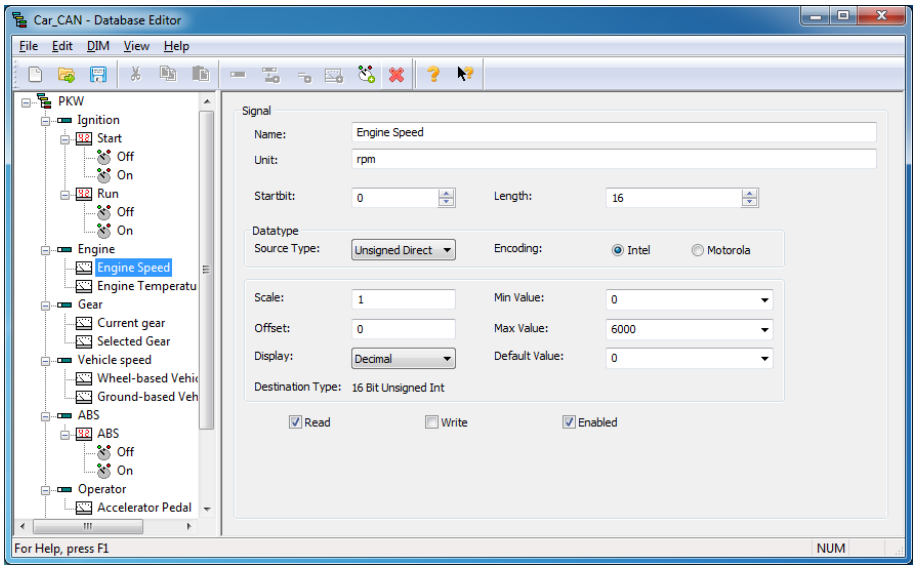
Ixxat®
BY HMS NETWORKS

DIMEditor

SOFTWAREHANDBUCH

DEUTSCH



HMS Technology Center Ravensburg GmbH
Helmut-Vetter-Straße 2
D-88213 Ravensburg
Germany

Tel.: +49 751 56146-0
Fax: +49 751 56146-29
Internet: www.hms-networks.com
E-Mail: info-ravensburg@hms-networks.com

Support

Sollten Sie zu diesem, oder einem anderen HMS Produkt Support benötigen, füllen Sie bitte das Supportformular auf www.ixxat.com/de/support aus.

Unsere internationalen Supportkontakte finden Sie im Internet unter www.ixxat.com

Copyright

Geschützte Warenzeichen

Handbuchnummer: 1.02.0133.30000-DIMEdit
Version: D-2.31

Inhaltsverzeichnis

Kapitel 1

Übersicht

1.1 Das Daten-Interpreter-Modul (DIM)

Das Daten-Interpreter-Modul ermöglicht die in einer CAN-Nachricht übertragenen Daten als physikalische Größen bzw. Zustände zu **interpretieren**.
Das Kernstück des Daten-Interpreter-Moduls bildet hierbei der **Daten-Interpreter-Server**. Dieser empfängt CAN-Telegramme und führt die Interpretation anhand einer vom Benutzer festgelegten Datenbasis durch. In der Datenbasis sind **Signaldefinitionen** enthalten, welche Länge und Position der für die Darstellung des **Signals** im CAN-Telegramm erforderlichen Bitfelder, sowie Format und anzuwendende Umrechnungsregeln beschreiben.
Das Daten-Interpreter-Modul unterstützt die folgenden Signaltypen:

- **Analoge Signale** in unterschiedlichen Kodierungen
- **Digitale Signale** mit zugeordneten Zuständen (bestimmt durch Wert und Name)

In der Interpretationsphase werden die entsprechenden Bitfelder eines CAN-Telegramms in Zahlenwerte umgewandelt und als physikalische Größen bzw. Zustände bereitgestellt.
Das Daten-Interpreter-Modul besteht aus drei Programmteilen (Abb. 1.1):

- Kernstück bildet der **Daten-Interpreter-Server**. Dieser empfängt CAN-Telegramme und führt die Interpretation anhand den in einer Datenbasis gespeicherten Interpretationsregeln durch.
- Mit dem **Datenbasis-Editor** kann der Benutzer Signalbeschreibungen und Interpretationsregeln erstellen, bearbeiten und speichern.
- Das **Signal-Modul** dient zum Anzeigen der Signalwerte.

Die nachfolgenden Kapitel vermitteln die Grundlagen für die Verwendung des Daten-Interpreter-Moduls. Im Kapitel "Bedienung - Quick Start" wird mit Hilfe des Datenbasis-Editors eine Beispiel-Datenbasis erstellt. Das Durcharbeiten dieses Kapitels ermöglicht einen Schnelleinstieg in das Daten-Interpreter-Modul.

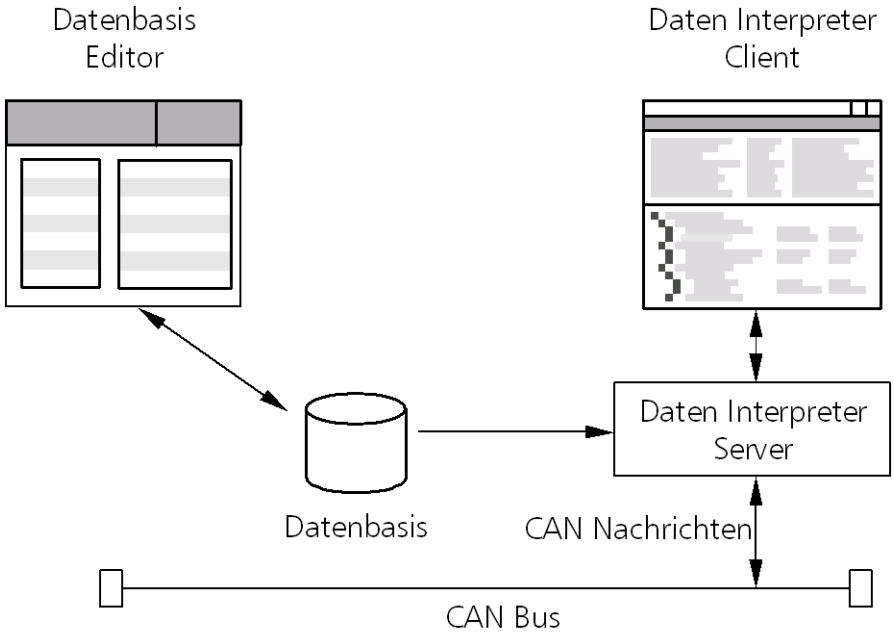


Abbildung 1.1: Komponenten des Daten-Interpreter-Moduls

Kapitel 2

Daten-Interpreter Grundlagen

2.1 Prinzip

In einem seriellen Bussystem werden Informationen in Form einer Folge von Bits (Bitstrom) übertragen. Erst eine zugehörige Interpretationsvorschrift gibt den übertragenen Bits eine Bedeutung und macht die übertragenen Informationen für den Empfänger nutzbar.
Grundsätzlich werden zwei Arten von Informationen übertragen:

- **Protokollinformationen** (beschreibt die Bedeutung der verbleibenden Information)
- **Nutzdaten**

Protokollinformation wird im Daten-Interpreter-Modul durch sogenannte **Daten-Interpreter-Multiplexer**, Daten werden durch **Signaldefinitionen** beschrieben. Beide Informationsarten werden im Daten-Interpreter-Modul in einer Datenbasis verwaltet.
Die zu interpretierende Nachricht wird (über eine Nachrichtendefinition) anhand des CAN-Identifiers ausgewählt. Die Position eines **Datenobjekts** (Daten-Interpreter-Multiplexer bzw. Signaldefinition) innerhalb dieser Nachricht wird durch die Angabe von **Anfangsbit** (Position des ersten Bits) und **Länge des Bitfeldes** beschrieben.
Im weiteren wird nun die Definition von Signalen sowie Multiplexern erläutert.

2.2 Definitionen von Signalen

Eine **Signaldefinition** ordnet einem Bitfeld innerhalb einer Nachricht (Bitstrom) eine bestimmte Bedeutung zu. Ein Signal ist durch die folgenden Attribute definiert:

- Name
- Bereich (Anfangsbit und Länge)
- Kodierung
  - Byte-Order (INTEL/Motorola)
  - Datentyp
- Umrechnungsregel: Derzeit wird eine lineare Umrechnung über Skalierungsfaktor und Offset unterstützt.

Mögliche Datentypen sind:

- Binärzahlen ohne Vorzeichen (Unsigned Integer, 1 - 64 Bit)
- Binärzahlen mit Vorzeichen (2 - 64 Bit) (Signed Integer, 1er Komplement, 2er Komplement)
- Gepackte BCD-Zahlen ohne Vorzeichen (4 - 64 Bit, 4 Bit Granularität)
- Gepackte BCD-Zahlen mit Vorzeichen (8 - 64 Bit, 4 Bit Granularität)
- IEEE Fließkommazahlen (32 bzw. 64 Bit)

Um eine Vorbelegung bzw. eine Überprüfung der Signalwerte zu ermöglichen, sind weitere Angaben, über

- den Minimal-, Maximal- und Standard-Wert, sowie den
- Zugriffstyp des Signals (lesbar und/oder schreibbar) nötig.

Weitere Einstellungen werden verwendet, um die Ausgabe der Signale über das Signal-Modul festzulegen. Dies sind:

- Farbe
- Ausgabeformat (ASCII, Binär, Oktal, Dezimal, Hexadezimal)

2.2.1 Byte-Order (Endianess)

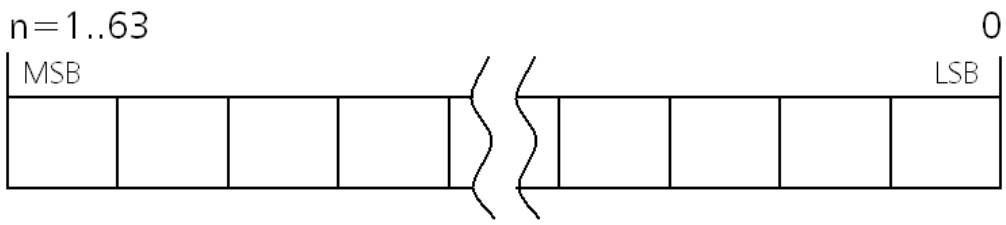
Das Byte-Order-Flag legt die verwendete Endianess (Intel oder Motorola) fest. Dies wirkt sich auf die Richtung in der die Signalbits extrahiert werden und auf die Position des Startbits aus. Die Startbitposition wird für Intel-kodierte Signale in "Intel standard" Notation und für Motorola-kodierte Signale in "Motorola forward MSB" Notation angegeben.
Die Startbitposition von Motorola-Signalen wird in "Motorola forward MSB" Notation festgelegt. Das Layout ist dasselbe wie bei "Motorola forward LSB" aber das Startbit und das Endebit sind vertauscht. In der "Motorola forward MSB" Notation ist die Startbitposition immer niedriger als die Endebitposition.
Intern wird bei vielen Tools die "Motorola forward LSB" Notation verwendet, hier ist die Startbitposition höher als die Endbitposition.

2.2.2 Digitale Signale

Digitale Signale unterscheiden sich von analogen Signalen durch zusätzlich zugeordnete Zustandsbeschreibungen.
Wichtig: Die Zuordnung von Zuständen ist nur für Signale des Datentyps "Vorzeichenlose Binärzahl" sinnvoll.

2.2.3 Unterstützte Datentypen

Binärzahl ohne Vorzeichen



Eine Binärzahl ohne Vorzeichen wird als Betrag angesehen.

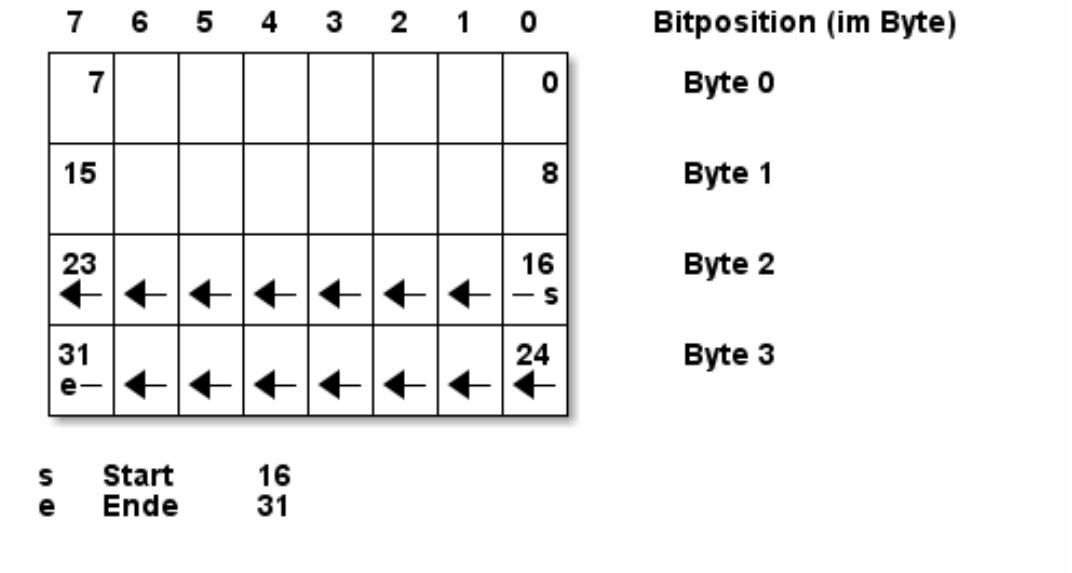


Abbildung 2.1: Für Intel-Signale liegt das Startbit am LSB.

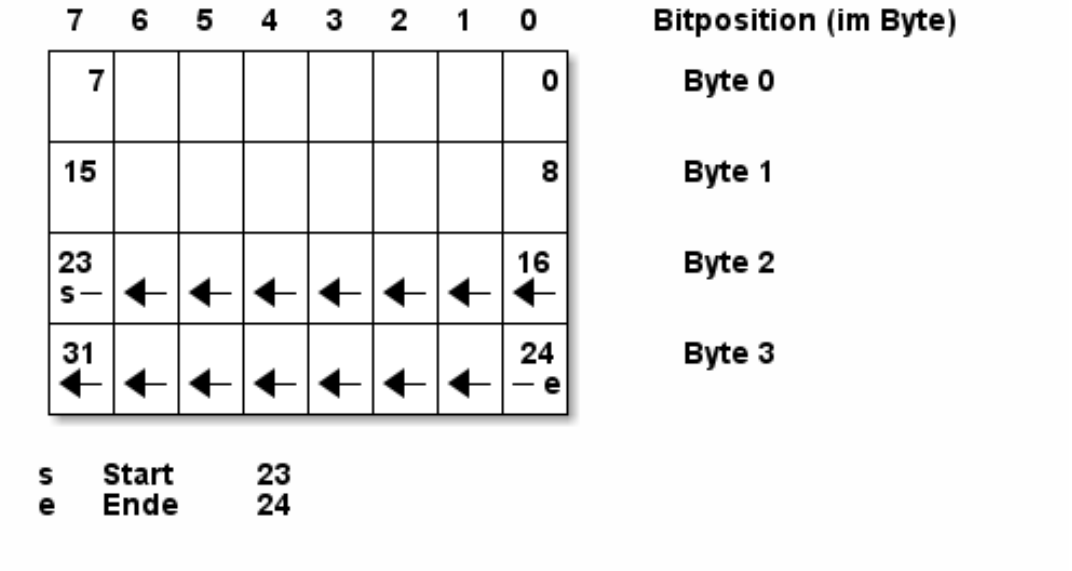


Abbildung 2.2: "Motorola forward MSB" Notation

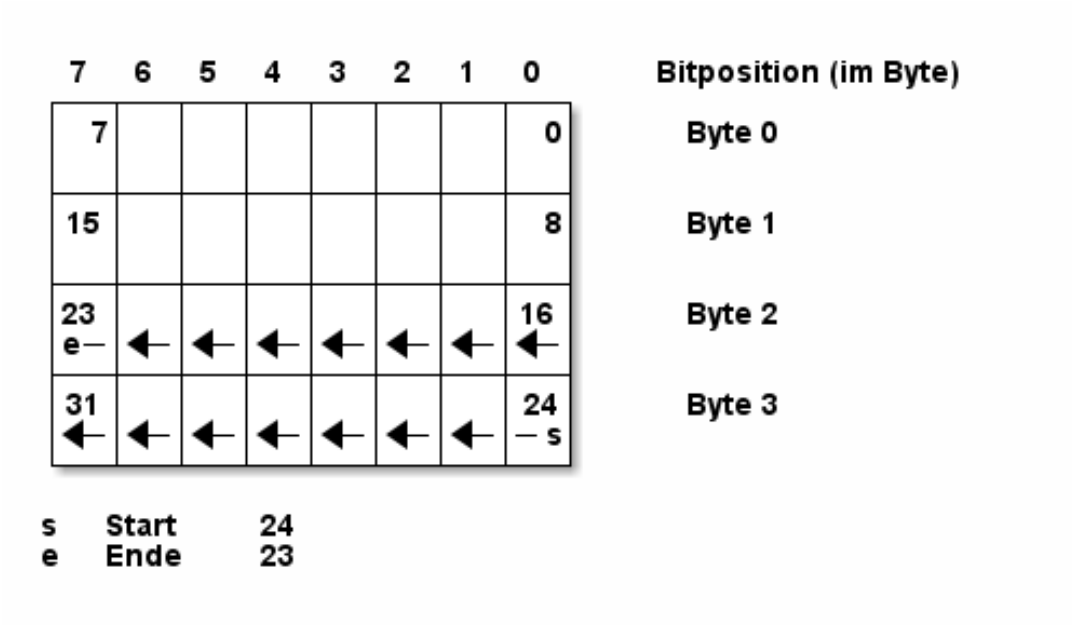
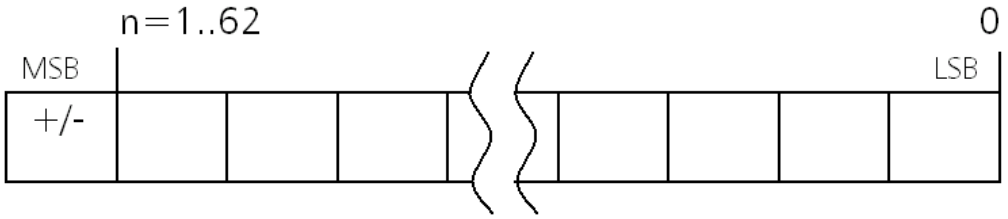


Abbildung 2.3: "Motorola forward LSB" Notation

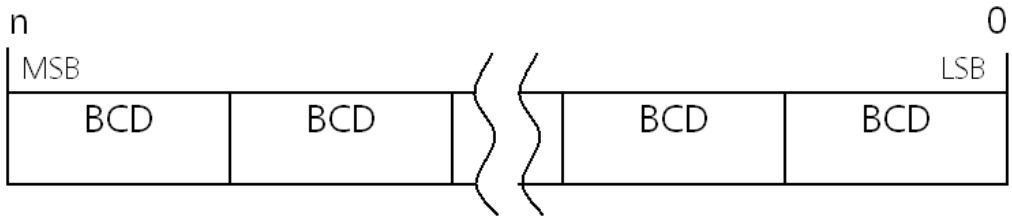
Binärzahl mit Vorzeichen



Bei den Binärzahlen mit Vorzeichen wird angenommen, dass das Vorzeichenbit im MSB abgelegt ist.
Es existieren drei unterschiedliche Kodierungen:

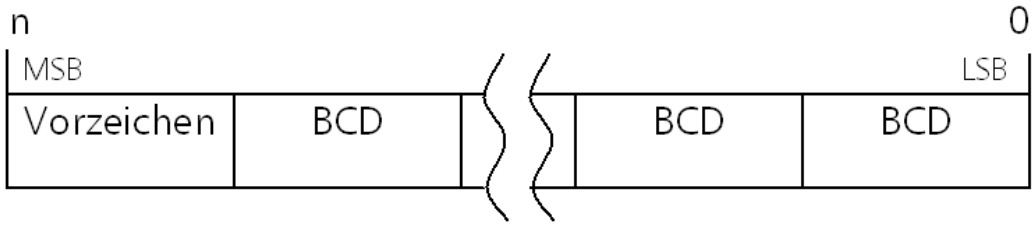
- Betrag mit Vorzeichenbit
- 1er Komplement
- 2er Komplement

Unsigned BCD (4 - 64 Bit, 4 Bit Granularität)



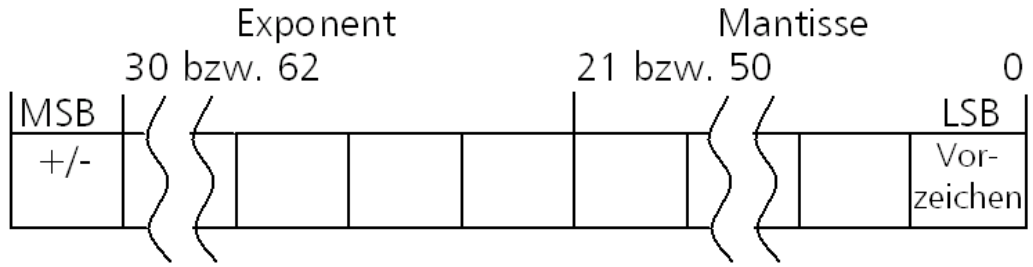
Jeweils vier Bits werden zu einer BCD-Ziffer von 0 bis 9 zusammengefasst
Die maximal mögliche Anzahl an BCD Ziffern richtet sich nach der Telegrammlänge.

Signed BCD (8 - 64 Bit, 4 Bit Granularität)



Jeweils vier Bits werden zu einer BCD-Ziffer von 0 bis 9 zusammengefasst.
Das Vorzeichen besteht ebenfalls aus einem 4 Bit-Block mit dem Wert 0000 für ein positives Vorzeichen und 1000 (8 hex) für ein negatives Vorzeichen.
Die maximal mögliche Anzahl an BCD Ziffern richtet sich nach der Telegrammlänge.

IEEE Floating Point (32 oder 64 Bit)



IEEE Fließkomma kodierte Zahlen bestehen aus einem Vorzeichenbit, einer Mantisse und einem Exponenten. Es werden zwei Formate unterstützt:

- IEEE 32 Bit (22 Bit Mantisse, 7 Bit Exponent)
- IEEE 64 Bit (51 Bit Mantisse, 10 Bit Exponent)

ASCII (8 - 64 Bit, 8 Bit Granularität)

Die Daten werden als ASCII-String interpretiert.

2.3 Gegenüberstellung der Integer-Datentypen

Value	Unsigned binary	Signed integer	1's compl.	2's compl.	Unsigned BCD	Signed BCD
+15	1111	-	-	-	-	-
+14	1110	-	-	-	-	-
+13	1101	-	-	-	-	-
+12	1100	-	-	-	-	-
+11	1011	-	-	-	-	-
+10	1010	-	-	-	-	-
+9	1001	-	-	-	1001	0000 1001
+8	1000	-	-	-	1000	0000 1000
+7	0111	0111	0111	0111	0111	0000 0111
+6	0110	0110	0110	0110	0110	0000 0110
+5	0101	0101	0101	0101	0101	0000 0101
+4	0100	0100	0100	0100	0100	0000 0100
+3	0011	0011	0011	0011	0011	0000 0011
+2	0010	0010	0010	0010	0010	0000 0010
+1	0001	0001	0001	0001	0001	0000 0001
+0	0000	0000	0000	0000	0000	0000 0000
-0	-	1000	1111	0000	-	1000 0000
-1	-	1001	1110	1111	-	1000 0001
-2	-	1010	1101	1110	-	1000 0010
-3	-	1011	1100	1101	-	1000 0011
-4	-	1100	1011	1100	-	1000 0100
-5	-	1101	1010	1011	-	1000 0101
-6	-	1110	1001	1010	-	1000 0110
-7	-	1111	1000	1001	-	1000 0111
-8	-	-	-	1000	-	1000 1000
-9	-	-	-	-	-	1000 1001
-10	-	-	-	-	-	-
-11	-	-	-	-	-	-
-12	-	-	-	-	-	-
-13	-	-	-	-	-	-
-14	-	-	-	-	-	-
-15	-	-	-	-	-	-

2.4 Daten-Interpreter-Multiplexer

Ein **Daten-Interpreter-Multiplexer** (Protokollinformation) ist ein Bitfeld innerhalb des Bitstroms, das die Bedeutung der im Bitstrom enthaltenen Information spezifiziert. Ein Daten-Interpreter-Multiplexer ist durch eine Multiplexer-Maske und einen Multiplexer-Wert bestimmt:

- Die **Multiplexer-Maske** beschreibt über Anfangsbit und Länge, wo im Bitstrom sich das Bitfeld der Multiplexermaske befindet
- Der eigentliche Wert des Multiplexers wird durch den **Multiplexer-Wert** bzw. einen **Alternativ-Wert (ElseValue)** angegeben.

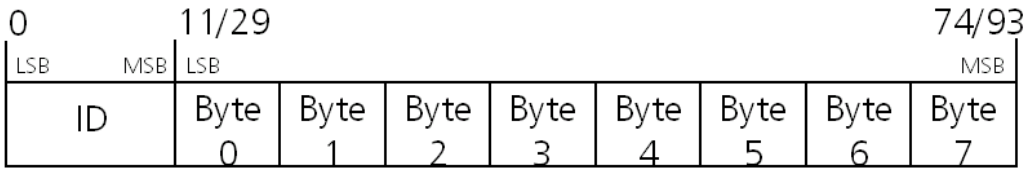


Abbildung 2.4: Aufbau eines CAN-Schicht 2-Nachricht im Daten-Interpreter-Modul

Beispiel

Angenommen, ein Protokoll überträgt im ersten Byte des Datenfeldes eine Funktionsnummer, welche die Bedeutung der restlichen Daten-Bytes im Telegramm bestimmt.
Um dieses Protokoll mit dem Daten-Interpreter-Modul zu interpretieren, wird eine Multiplexer-Maske (Bereich) über das erste Byte festgelegt. Mit Hilfe von zugeordneten Multiplexer-Werten können jetzt die relevanten Funktionsnummern beschrieben werden.
Zu jeder Funktionsnummer (Multiplexer-Wert) werden Signaldefinitionen zugeordnet und damit die Bedeutung der übertragenen Daten festgelegt.

2.4.1 Alternativ-Wert (ElseValue)

Zu jeder Multiplexer-Maske ist in der Daten-Interpreter-Datenbasis ein Alternativ-Zweig **(ElseValue)** vorgesehen.
Die zu einem ElseValue gehörenden Interpretationsregeln werden angewendet, wenn kein in der Datenbasis enthaltener Multiplexer-Wert zutrifft.
Unterhalb eines ElseValues sind keine weiteren Multiplexer mehr zugelassen.

2.4.2 Multiplexerunabhängige Signaldefinitionen

Multiplexerunabhängige Signaldefinitionen nehmen eine Sonderstellung innerhalb der Daten-Interpreter-Datenbasis ein, da sie für jede empfangene Nachricht neu angewendet werden.
Ein Beispiel eines multiplexerunabhängigen Signals ist z.B. ein Protokoll, welches statische Informationen, wie z.B. die Adresse des Zielknotens immer im selben Bereich einer Nachricht ablegt.

2.5 Daten-Interpreter-Modul und das CAN-Protokoll

Das Daten-Interpreter-Modul behandelt CAN-Nachrichten wie eine fortlaufende Bitfolge, bestehend aus CAN-Identifier und Datenfeld, d.h. macht bei der Interpretation keinen Unterschied zwischen ID- und Datenbereich. Abb. 2.4 zeigt den Aufbau einer CAN-Schicht 2-Nachricht aus der Sicht des Daten-Interpreter-Moduls.
Die CAN-Nachricht besteht aus einem 11(29) Bit breiten Identifier und einem 64 Bit (8 Byte) breiten Datenfeld. Theoretisch kann man diese zusammengenommen 75(93) Bits nutzen, um beliebige Bitfolgen zu übertragen.

2.6 Interpretation empfangener CAN-Telegramme

Die Interpretation durchläuft im wesentlichen die folgenden Schritte (Abb. 2.5):

- Eine CAN-Nachricht wird empfangen und zur Interpretation übergeben
- Multiplexerunabhängige Signale werden aus der Nachricht extrahiert und interpretiert
- All Nachrichtenbeschreibungen werden der Reihe nach auf Übereinstimmung geprüft. Wenn eine Nachrichtenbeschreibung per ID oder Maske/Wert-Paar übereinstimmt, werden folgende Schritte übernommen:
  - Die erste Mux-Maske wird über das Telegramm gelegt und der Mux-Wert bestimmt
  - Ist der entsprechende Mux-Wert nicht definiert, so wird der zugeordnete Else-Zweig zur Interpretation benutzt.
  - Die zum aktuellen Mux-Wert bzw. Else-Zweig gehörigen Signale werden aus dem Telegramm extrahiert und interpretiert.
  - Anschließend wird der nächste Submultiplexer bestimmt und rekursiv die Interpretation durchgeführt.
- Dieselben Schritte werden auf jede Nachricht angewandt, wenn eine globale Muxmaske definiert ist.

Der Interpretationsvorgang kann auch anhand einer Baumstruktur dargestellt werden (Abb. 2.6).
Auf der Wurzelebene befindet sich das Projekt mit den dazu gehörenden Projektdaten.
Unterhalb der Wurzel sind die Nachrichtenobjekte angeordnet.
Eine Ebene tiefer befinden sich die multiplexerunabhängigen Signale sowie die erste Multiplexer-Maske.
Die darunter liegende Multiplexer-Maske (MuxMaske) beschreibt zusammen mit den Multiplexer-Werten (MuxValues) die Multiplexer.
Der Alternativ-Zweig (ElseValue) wird zur Interpretation herangezogen, wenn keiner der MuxValues auf gleicher Ebene zutrifft.
Unterhalb eines MuxValues können sich weitere Multiplexer (Mask-Value-Paare) befinden.

2.7 Umrechnungsvorschrift für analoge Signale

Analoge Signale werden anhand einer linearen Umrechnungsvorschrift von einem Rohwert in einen physikalischen Wert umgerechnet.
Abb. 2.7 zeigt den Ablauf des Umrechnungsvorgangs und die Parameter, die dabei benutzt werden.

Beispiel

In einem CAN-Telegramm wird der von einem Wärmesensor gelieferte Messwert übertragen.
Der Wärmesensor liefert Werte zwischen 0 und 4095 (12 Bit AD-Wandler, 2^12 - 1 = 4095). Dies entspricht Temperaturen zwischen -30 C und +120 C (Abb. 2.8).
Die Werte für Scale und Offset beschreiben die Steigung und Nullpunkt-Verschiebung der Umrechnungsgeraden und ergeben sich zu:

```
Scale = (PhysMax - PhysMin) / (LogMax - LogMin)
Offset = PhysMin
```

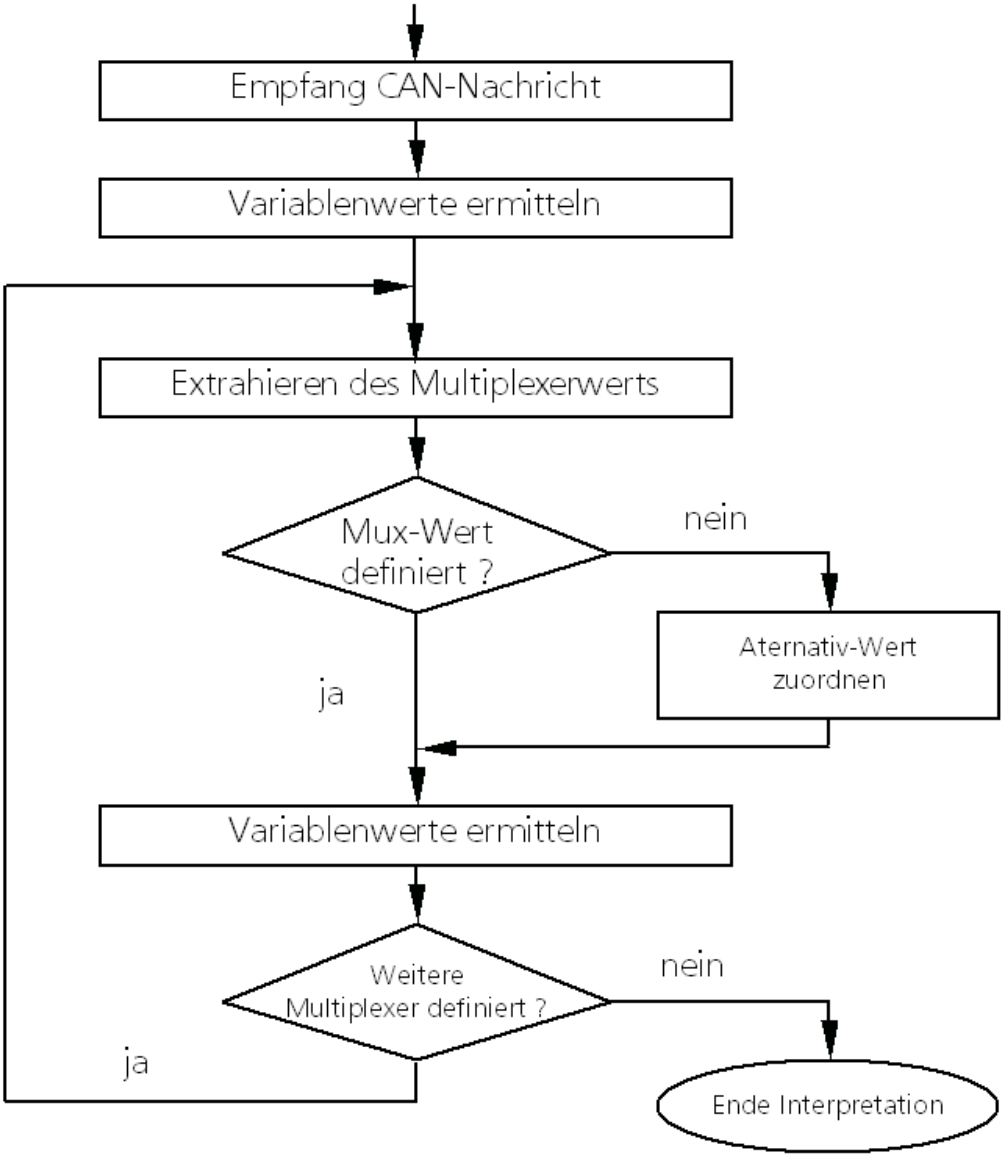


Abbildung 2.5: Ablauf der Interpretation von Nachrichten

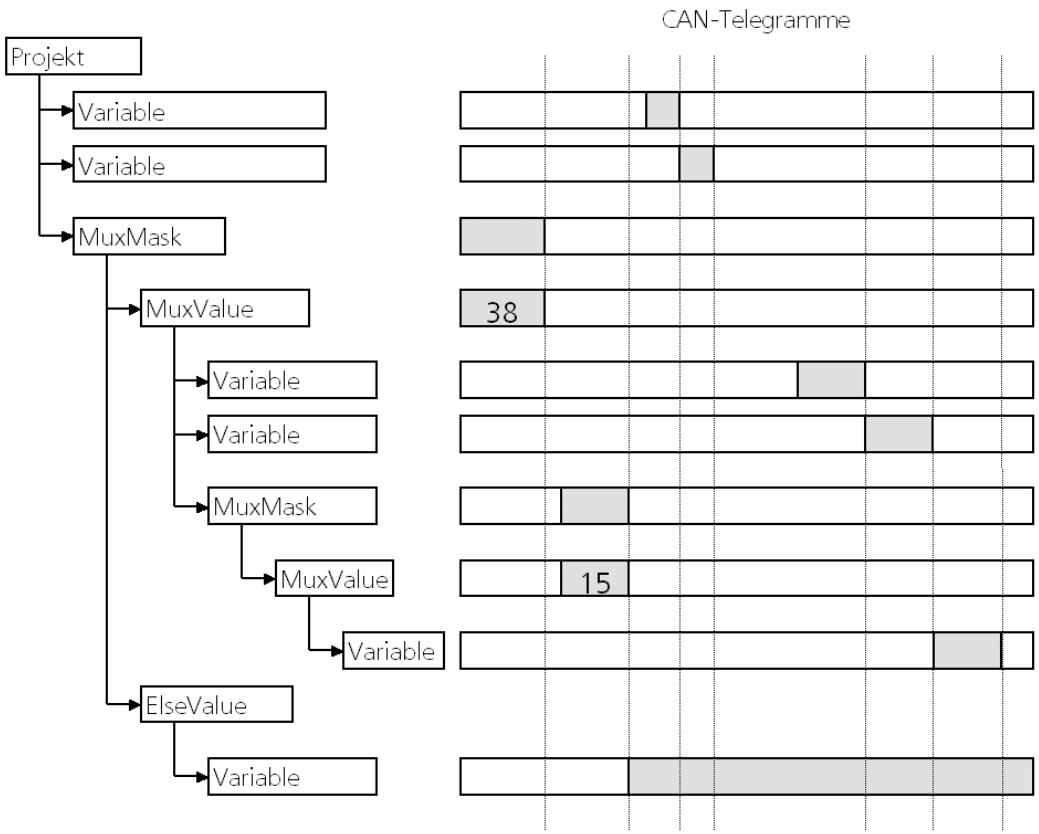


Abbildung 2.6: Struktur einer Daten-Interpretations-Datenbasis

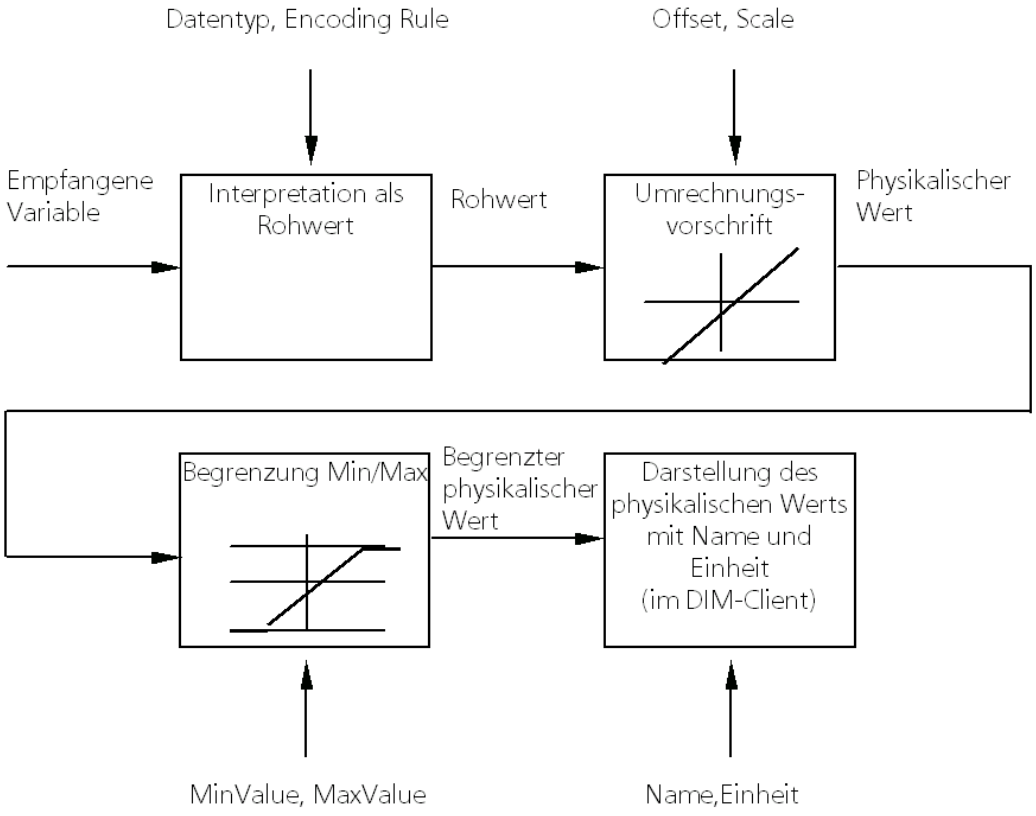


Abbildung 2.7: Umrechnung von Signalen

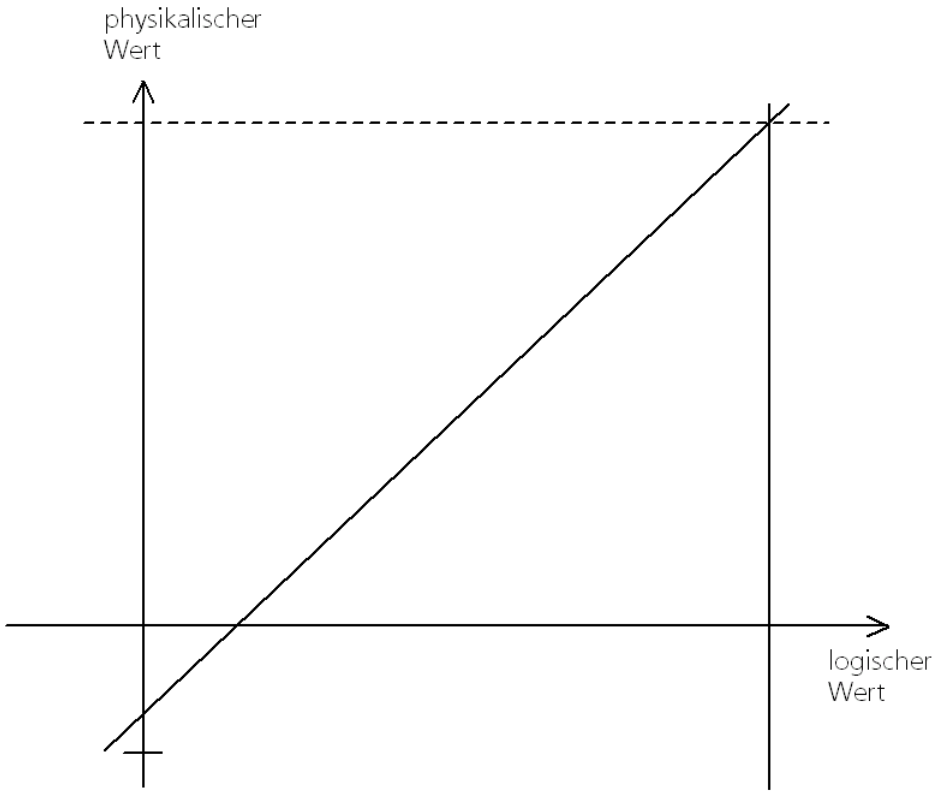


Abbildung 2.8: Lineare Umrechnung zwischen logischem und physikalischem Wert

Für das in Abb. 2.8 gezeigte Beispiel ergibt sich:

- Scale = 150 °C / 4095 Einheiten = 0,03663 °C / Einheit
- Offset = -30 °C

Kapitel 3

Bedienung des Datenbasis-Editors

3.1 Übersicht

Mit dem Datenbasis-Editor erfolgt die Eingabe der Interpretationsregeln für das Daten-Interpreter-Modul.
Generell können bei allen Eingabefeldern, welche ganzzahlige Werte verlangen diese sowohl dezimal oder hexadezimal mit vorangestelltem "0x" eingegeben werden.

3.2 Hauptfenster

Das Hauptfenster (Abb. 3.1) des Datenbasis-Editors besteht im wesentlichen aus zwei Teilen. Links wird eine Baumansicht der aktuellen Datenbasis dargestellt.
Rechts können die Informationen zum aktuell ausgewählten Baum-Knoten eingesehen und editiert werden. Je nach gewähltem Knotentyp (**Datensatz-Typ**) ändert sich das rechts dargestellte Formular.

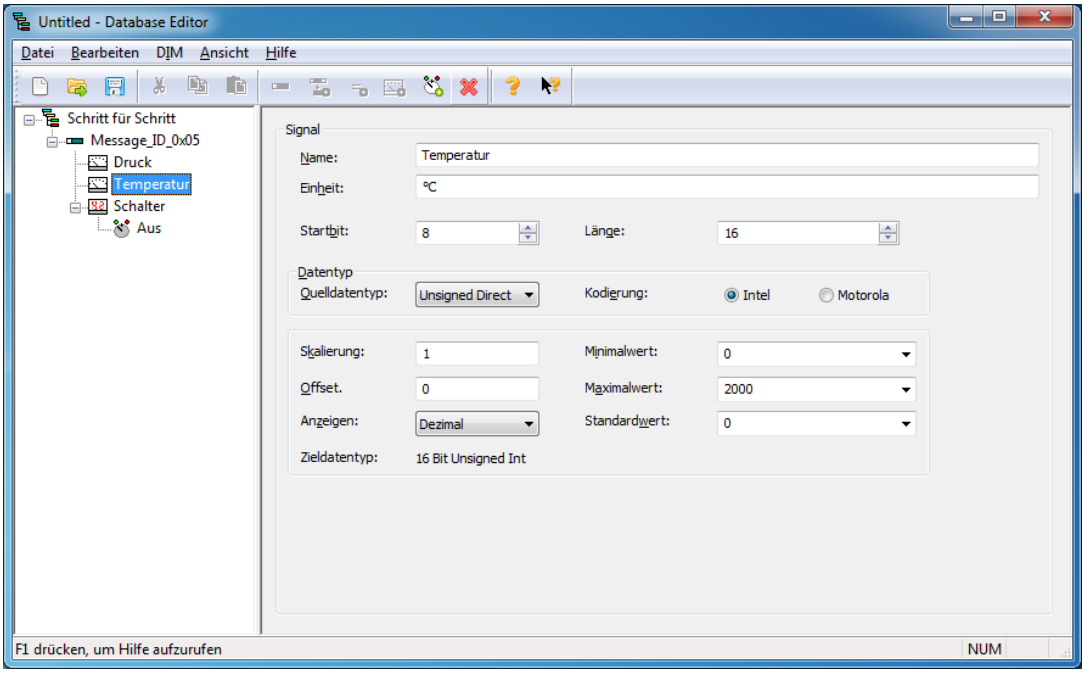


Abbildung 3.1: Das Hauptfenster des Datenbasis-Editors

3.3 Datensätze in der Daten-Interpreter-Datenbasis

In einer Daten-Interpreter-Datenbasis kommen sieben verschiedene Arten von Datensätzen vor:

- Projekt
- Nachricht
- MuxMaske (Bitposition/Beschreibung eines Multiplexers)
- MuxValues (enthalten zu den Multiplexern zugehörige Werte)
- ElseValues (Alternativ-Zweig)
- Signal-Definition (Analoge Signale)
- State-Definitionen (Zustandsbeschreibung digitaler Signale)

3.3.1 Projekt

Der Projekt-Datensatz befindet sich auf der Wurzelebene. Er enthält alle projektspezifischen Daten. Dazu gehören:

Attribut	Bemerkung
Projektname	max. 63 Zeichen
Kunde max.	63 Zeichen
Benutzer	max. 63 Zeichen
Version	max. 63 Zeichen
Kommentar	max. 1023 Zeichen
Datum	max. 63 Zeichen (wird beim Speichern aktualisiert)
Frame-Type	11 oder 29-Bit Identifier

Eine Ebene tiefer sind die folgenden Datensatz-Typen erlaubt:

- keine, eine oder mehrere Nachrichten
- keine, eine oder mehrere Signaldefinitionen
- höchstens eine MuxMaske

Die Eingabe der Projektdaten erfolgt über die Eingabemaske in Abb. 3.2.

3.3.2 Nachricht

In einer Nachrichtendefinition wird die zu interpretierende Nachricht über den CAN-Identifier und die Länge der Nachricht spezifiziert. Ein Nachrichtenobjekt kann als Nachrichtenfilter anhand des Identifiers und der Nachrichtenlänge (data length code, DLC) betrachtet werden.

Attribut	Bemerkung
Identifier	DWORD
Name	max. 63 Zeichen
Nachrichtenlänge (DLC)	DWORD
Zykluszeit	Auflösung in 100 ns

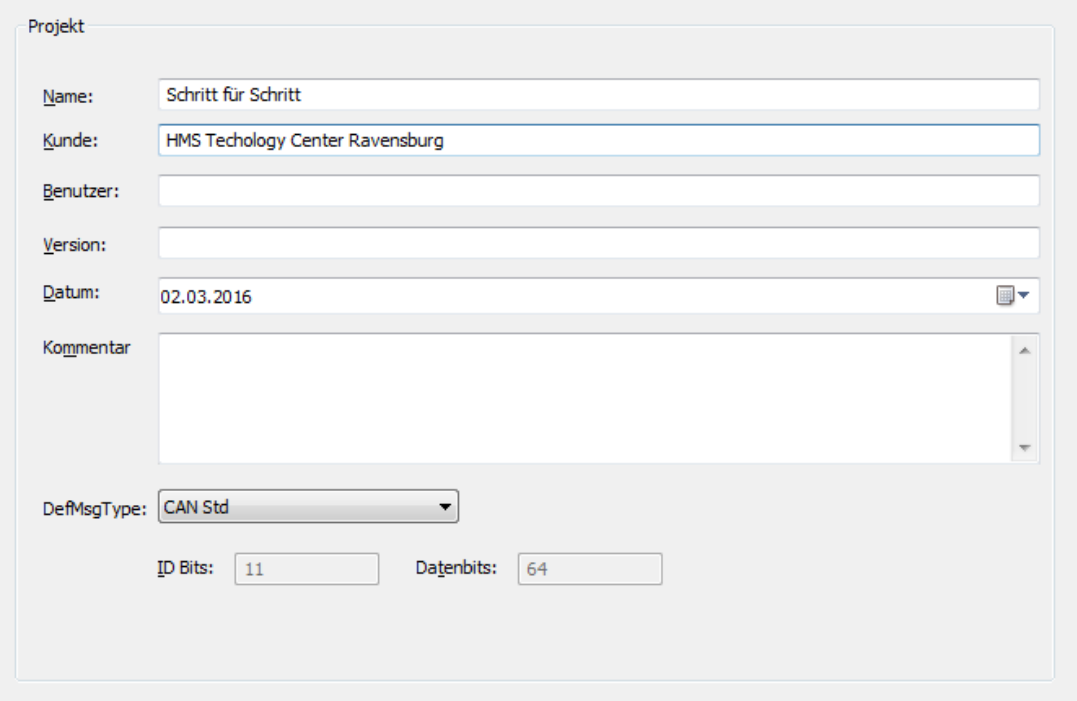


Abbildung 3.2: Eingabemaske für Projektdaten

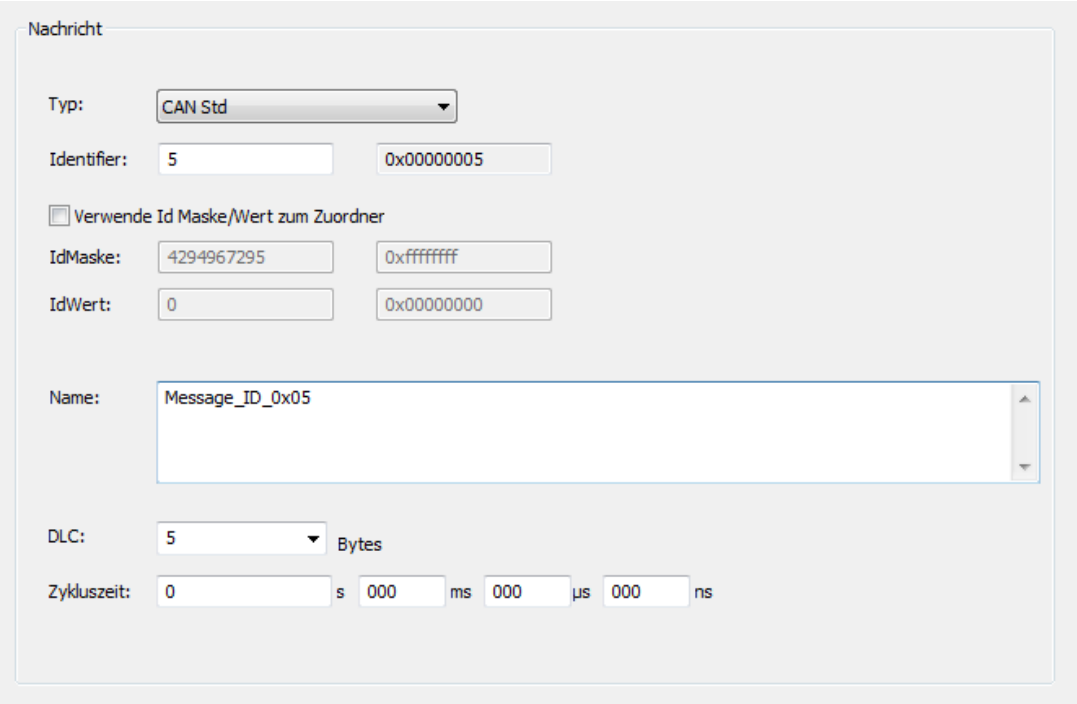


Abbildung 3.3: Eingabemaske für Nachrichtendaten

Eine Ebene tiefer sind die folgenden Datensatz-Typen erlaubt:

- keine, eine oder mehrere Multiplexer-unabhängige Signale
- höchstens eine MuxMaske

Die Eingabe der Nachrichtendaten erfolgt über die Eingabemaske in Abb. 3.3.

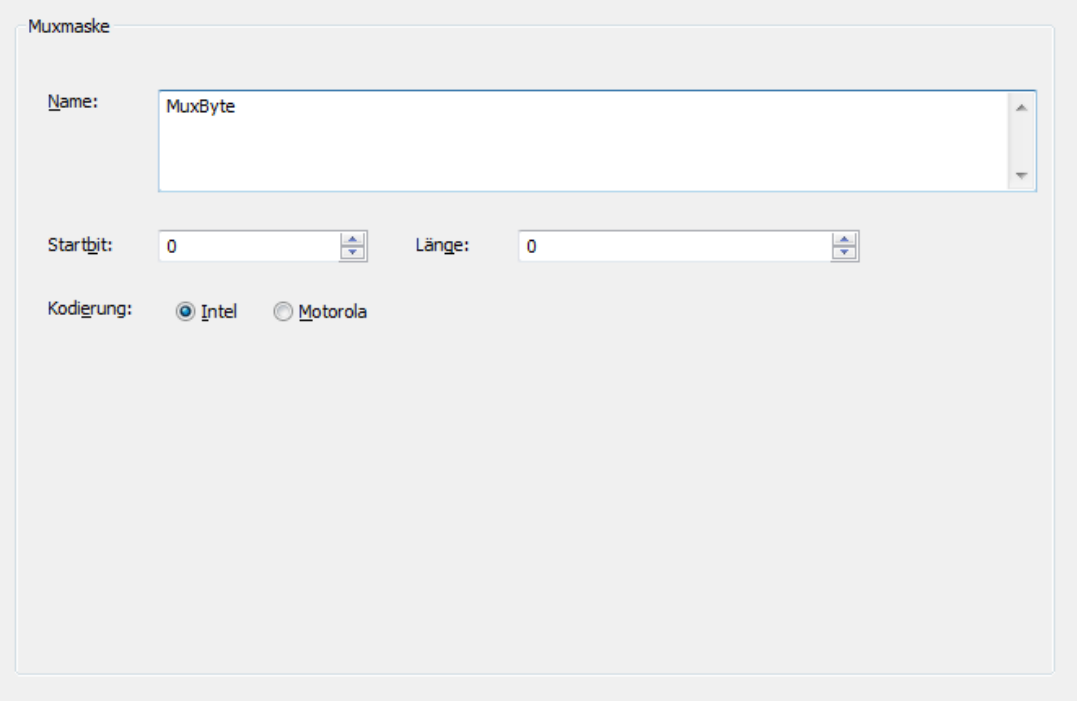


Abbildung 3.4: Eingabemaske für MuxMasken

3.3.3 MuxMaske

Eine MuxMaske beschreibt den Bereich innerhalb einer CAN-Nachricht in dem der zugehörige Multiplexer-Wert liegt.
Für die Beschreibung einer MuxMaske stehen folgende Felder zur Verfügung:

Attribut	Bemerkung
Name	max. 63 Zeichen
Anfangsbit	Integer [0..Telegrammlänge-1]
Länge	Integer [1..32]

Unterhalb einer MuxMaske sind die folgenden Datensatz-Typen erlaubt:

- kein, ein oder mehrere MuxValues
- ein ElseValue (Alternativ-Wert)

Die Dateneingabe erfolgt über die Eingabemaske in Abb. 3.4.

3.3.4 MuxValue

Ein MuxValue ist immer einem MuxMask-Datensatz zugeordnet. Er wird durch die folgenden Attribute beschrieben:

Attribut	Bemerkung
Name	max. 63 Zeichen
Wert	Integer

Ein MuxValue beschreibt den Wert eines Multiplexers. Das Attribut "Wert" dient als Vergleichswert bei der Interpretation von CAN-Nachrichten.
Unter einem MuxValue sind folgende Datensatz-Typen erlaubt:

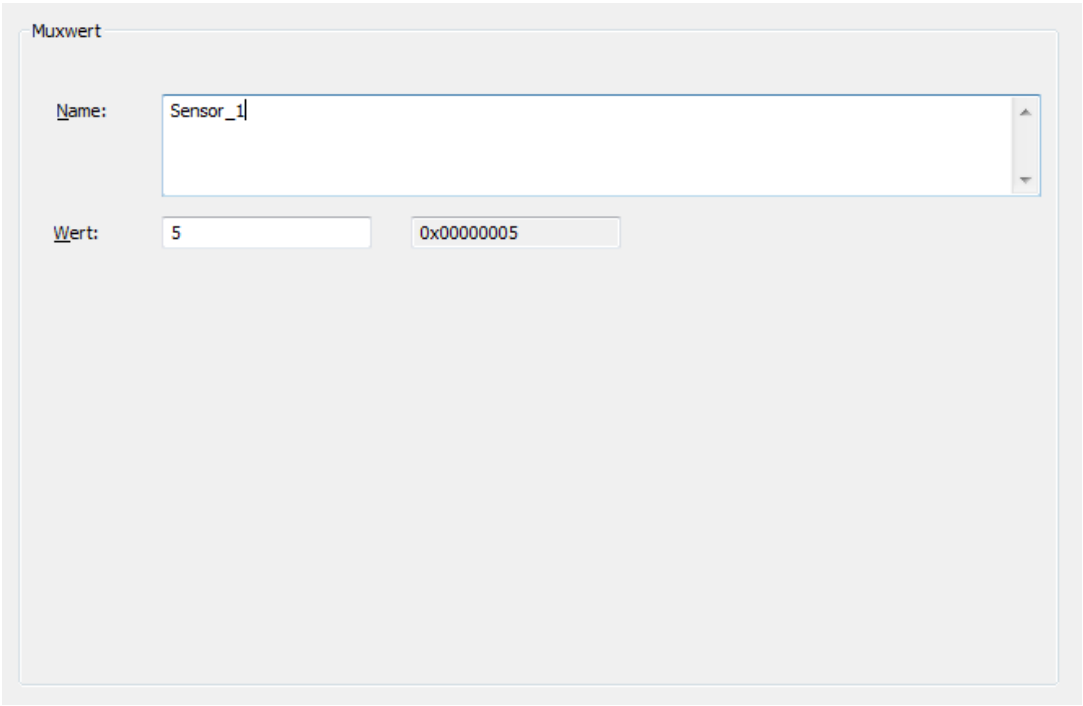


Abbildung 3.5: Eingabemaske für MuxValues

- keine, eine oder mehrere Signaldefinitionen
- höchstens eine MuxMaske

Die Dateneingabe erfolgt über die Eingabemaske in Abb. 3.5.

3.3.5 ElseValue

Der ElseValue steht für alle Werte eines Multiplexers die nicht explizit definiert sind.

Attribut	Bemerkung
Name	max. 63 Zeichen
Status	(wird vom Datenbasis-Editor nicht dargestellt)

Eine Ebene tiefer sind folgende Datensatz-Typen erlaubt:

- Signaldefinition

ElseValue-Knoten sind an die Verwendung mit MuxMasken gekoppelt und werden im Datenbasis-Editor deshalb automatisch angelegt bzw. gelöscht.
Die Dateneingabe erfolgt über die Eingabemaske in Abb. 3.6.

3.3.6 Signal-Definition

In einer Signaldefinition werden alle Attribute, die für die Beschreibung eines Signals notwendig sind, spezifiziert:

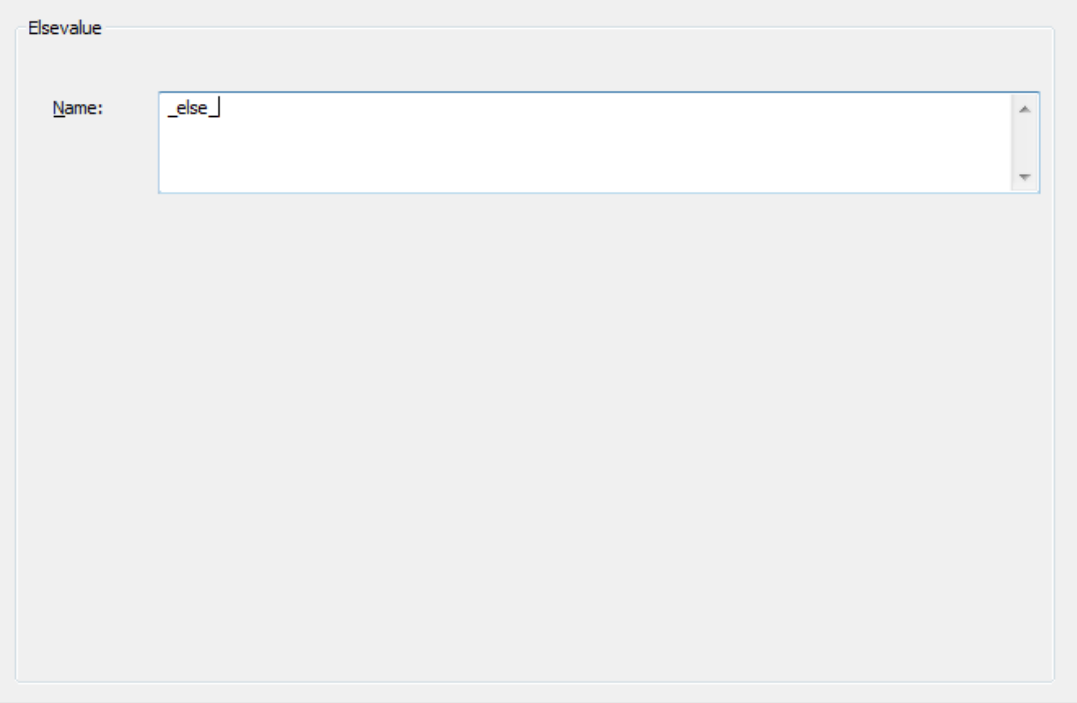


Abbildung 3.6: Eingabemaske für ElseValues

Attribut	Bemerkung
Name	max. 63 Zeichen
Einheit	max. 63 Zeichen
Anfangsbit	Integer [0..Telegrammlänge]
Länge in Bits	Integer [0..Telegrammlänge]
Skalierung	Double
Offset	Double
DefaultValue	DIM_VALUE
MinValue	DIM_VALUE
MaxValue	DIM_VALUE
Datentyp	DWORD
FormatString	max. 11 Zeichen (noch nicht unterstützt)
Farbe	DWORD (noch nicht unterstützt)
Status	DWORD (Read, Write, Enabled)

Die Funktionen der Attribute, die für die Umrechnung der Signale benutzt werden, werden im Kapitel "Umrechnungsvorschrift für analoge Signale" näher beschrieben.
Diese Attribute sind notwendig, damit der Daten-Interpreter-Server eine Übersetzung von Signalen aus einem Quelltyp in einen Zieltyp durchführen kann. Der Quelltyp eines Signals ergibt sich aus der Bitlänge, dem Attribut "Datentyp", sowie den Werten von Scale und Offset.
Die Attribute "Skalierung" und "Offset" werden für eine lineare Umrechnung der Signale benutzt.
Die Attribute "DefaultValue", "MinValue" und "MaxValue" sind vom Typ DIM_VALUE, das heißt ihre Bedeutung bzw. ihr Wert richtet sich nach dem Zieltyp des Signals.
Die Attribute FormatString und Farbe sind für zukünftige Versionen vorgesehen, werden momentan aber noch nicht unterstützt.
Im Attribut "Status" wird abgelegt, ob ein Signal lesbar, schreibbar bzw. überhaupt freigeschaltet ist.
Unterhalb einer Signaldefinition sind die folgenden Datensatz-Typen erlaubt:

- kein, ein oder mehrere States (Zustandsbeschreibungen)

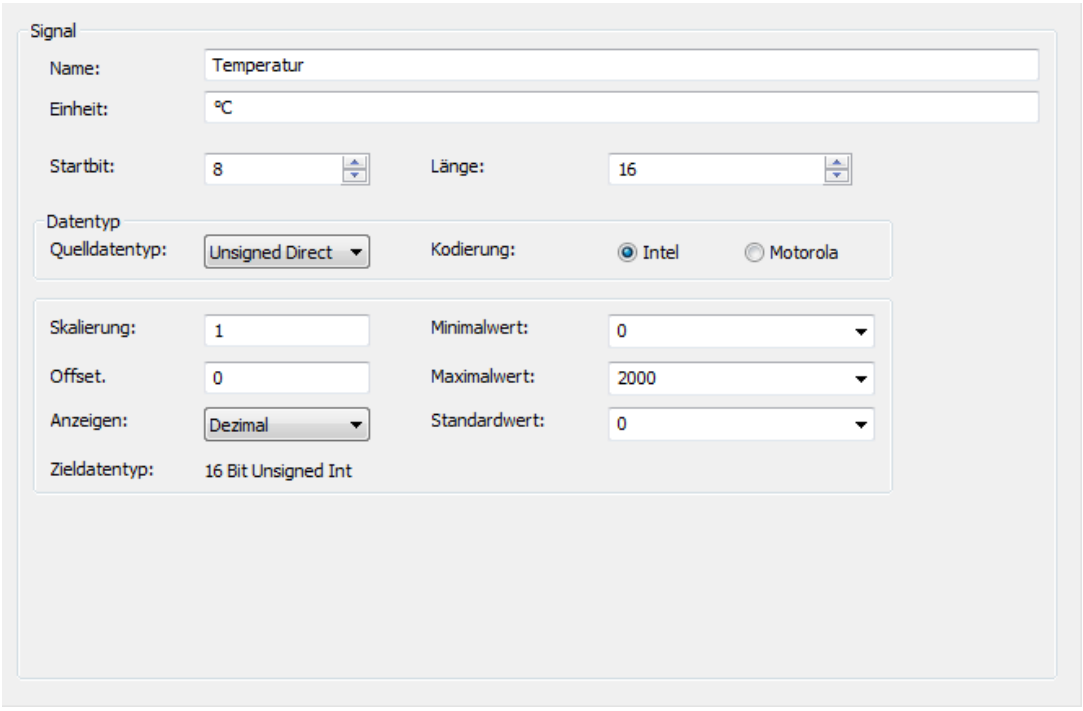


Abbildung 3.7: Eingabemaske für die Definition von Signalen

Die Dateneingabe erfolgt über die Eingabemaske in Abb. 3.7.

3.3.7 Zustandsbeschreibungen

Durch State-Datensätze werden die Zustände, die digitale Signale annehmen können, spezifiziert. Zustände sind durch eine Beschreibung (Name bzw. Bedeutung) sowie einem zugeordneten Wert beschrieben.
Sinnvoll sind States nur für Signale, die einen ganzzahligen Zieltyp haben.

Attribut	Bemerkung
Bedeutung	max. 63 Zeichen
Wert	DWORD

Unterhalb eines State-Datensatzes können sich keine weiteren Datensätze befinden.
Die Dateneingabe erfolgt über die Eingabemaske in Abb. 3.8.

3.4 Menüreferenz

Abb. 3.9 zeigt den Aufbau des Datenbasis-Editor-Menues.

3.4.1 File-Menü

Das File Menü verfügt über folgende Punkte:

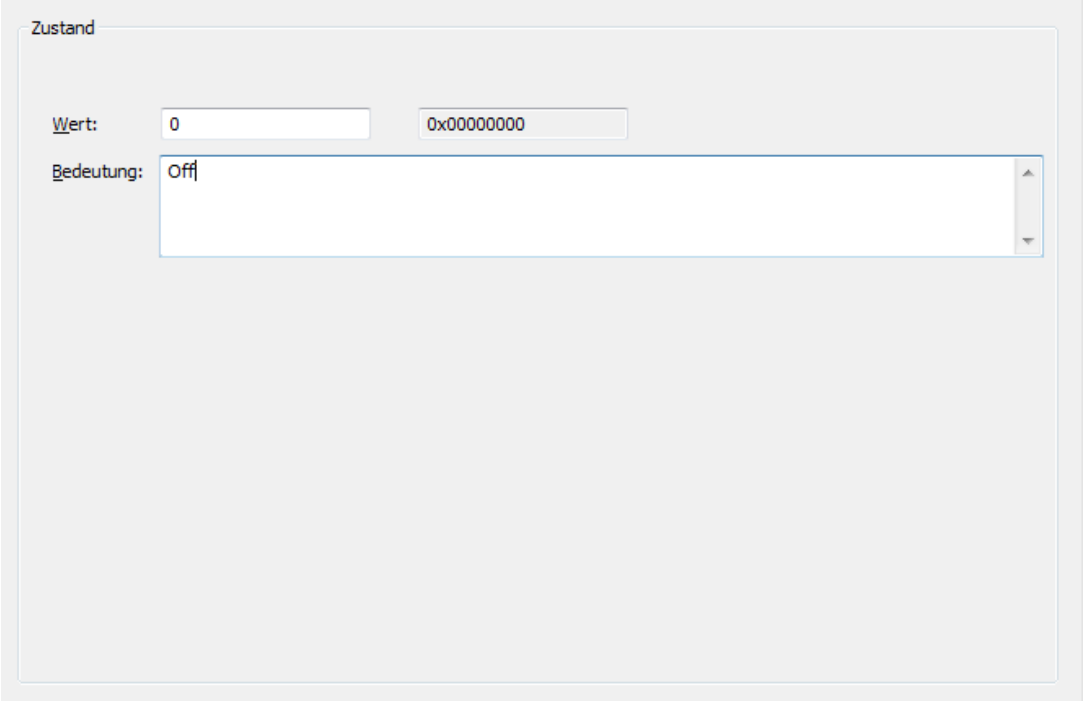


Abbildung 3.8: Eingabemaske für Zustände

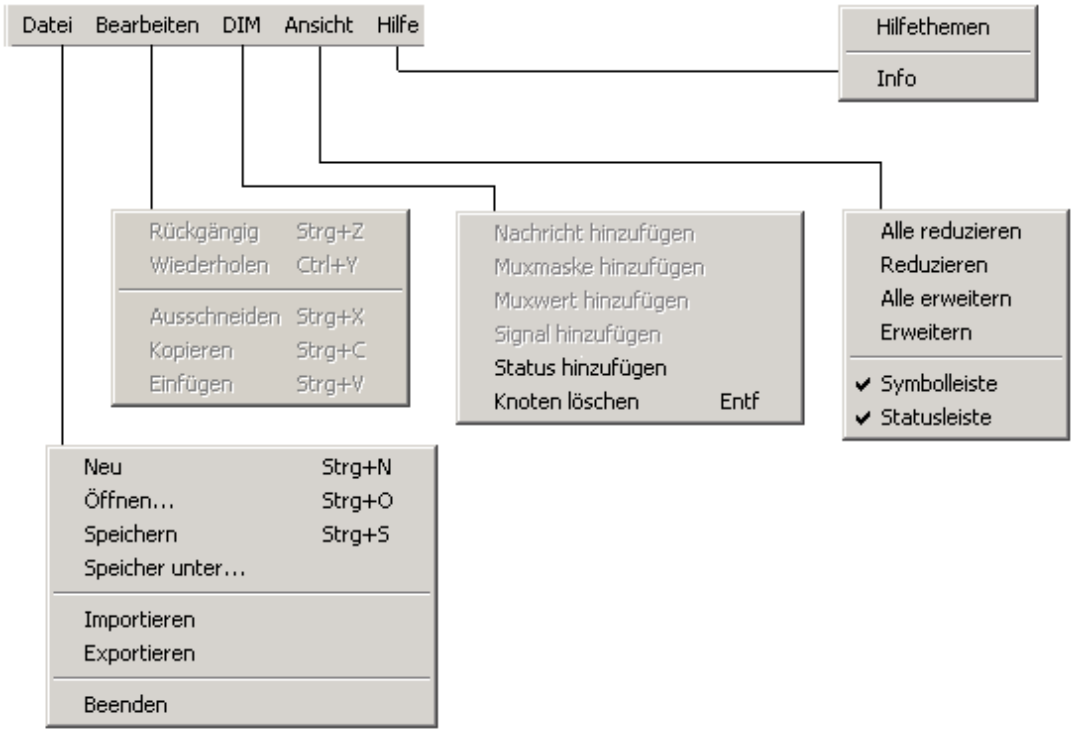


Abbildung 3.9: Menüfunktionen des Datenbasis-Editors

Menüpunkt	Funktion
New	Legt eine neue Projektdatenbasis an (Ctrl-N)
Open	Öffnet eine bereits vorhandene Projektdatenbasis (Ctrl-O)
Save	Speichert die Datenbasis unter dem aktuellen Namen (Ctrl-S)
Save As...	Speichert die Datenbasis unter einem anderen Namen
Import	Importiert die Datenbasis aus einem externen Format
Export	Exportiert die Datenbasis in ein externes Format
Recent File List	Enthält die zuletzt verwendeten Dateien
Exit	Beendet das Programm

3.4.2 Edit-Menü

Die Funktionen des Edit-Menüs beziehen sich auf das aktive Eingabefeld:

Menüpunkt	Funktion
Undo	Macht eine Änderung rückgängig (Ctrl-Z)
Cut	Schneidet markierten Text aus (Ctrl-X)
Copy	Kopiert markierten Text in die Zwischenablage (Ctrl-C)
Paste	Fügt Text aus der Zwischenablage ein (Ctrl-N)

3.4.3 DIM-Menü

Das DIM-Menü stellt Funktionen zum Einfügen/Löschen der Datenbasis-Datensätze zur Verfügung:

Menüpunkt	Funktion
Add Message	Fügt eine Nachricht zur Auswahl eines CAN-Identifiers unterhalb des Projektknotens ein
Add MuxMask	Fügt eine Multiplexer-Maske unterhalb des gewählten Datensatzes ein
Add MuxValue	Fügt einen Multiplexer-Wert unterhalb des gewählten Datensatzes ein
Add Signal	Fügt eine Signaldefinition unterhalb des gewählten Datensatzes ein
Add State	Fügt eine Zustandsdefinition unterhalb des gewählten Datensatzes ein
Delete Node	Löscht den aktuellen Knoten (Entf)

3.4.4 View-Menü

Das View-Menü stellt Funktionen zum Konfigurieren der Ansicht zur Verfügung.

Menüpunkt	Funktion
Collapse All	alle Knoten reduzieren
Collapse	markierten Knoten reduzieren
Expand All	alle Knoten erweitern
Expand	markierten Knoten erweitern
Toolbar	Blendet die Werkzeugleiste ein/aus
Statusbar	Blendet die Statuszeile ein/aus

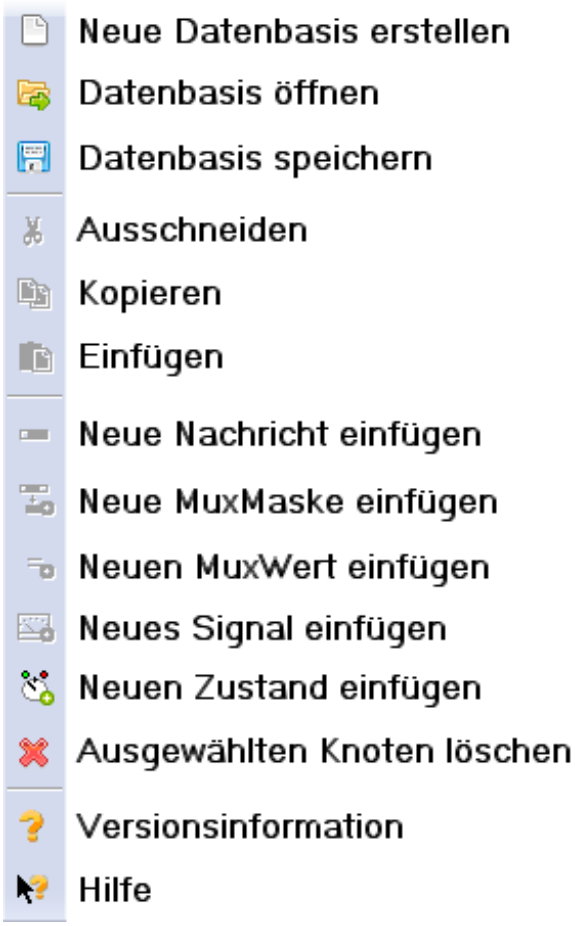


Abbildung 3.10: Schalterleiste Datenbasis-Editors

3.4.5 Help-Menü

Das Help-Menü enthält Funktionen zum Aufrufen der Hilfe und des About-Dialogs.

Menüpunkt	Funktion
Help Topics	Zeigt die Hilfedatei an (F1)
About DIMedit	Zeigt den About-Dialog an

3.5 Schalterleiste

Die wichtigsten Funktionen des Datenbasis-Editors sind auch über die Schalterleiste (Abb. 3.10) aufrufbar:

3.6 Zwischenablage

Die Zwischenablage ist ein Speicherbereich, der den einfachen Austausch von DIM-Daten (Multiplexer-Maske, Signal-Definition und andere Objekte) innerhalb des Datenbasis-Editors oder zwischen mehreren Datenbasis-Editoren in einem zweistufigen Vorgang ermöglicht. Dabei werden markierte Knoten vom Ursprungsort durch die Funktionen "Kopieren" oder "Ausschneiden" (engl. Copy/Cut) in die Zwischenablage kopiert. Diese werden durch die Funktion "Einfügen" (engl. Paste) aus der Zwischenablage in die Baumstruktur eines Datenbasis-Editors eingefügt. Das Zwischenablage-Menü verfügt über folgende Punkte:

Menüpunkt	Funktion
Cut	Schneidet markierten Knoten aus und kopiert ihn in die Zwischenablage (Ctrl-X)
Copy	Kopiert markierten Knoten in die Zwischenablage (Ctrl-C)
Paste	Fügt Knoten aus der Zwischenablage ein (Ctrl-V)

3.6.1 Cut-Menüpunkt

Das Ausführen des Cut-Menüpunktes bewirkt, dass der markierte DIM-Knoten in die Zwischenablage kopiert und anschließend aus der Baumstruktur gelöscht wird. Ausgeschnitten werden können alle DIM-Knoten/Objekte (mit Ausnahme des Projekt-Knotens).

3.6.2 Copy-Menüpunkt

Das Ausführen des Copy-Menüpunktes bewirkt, dass der markierte DIM-Knoten in die Zwischenablage kopiert wird. Kopiert werden können alle DIM-Knoten/Objekte.

3.6.3 Paste-Menüpunkt

Das Ausführen des Paste-Menüpunktes bewirkt, dass an den markierten DIM-Knoten der Inhalt der Zwischenablage angehängt wird. Wenn der Inhalt der Zwischenablage nicht zum markierten DIM-Knoten passt, wird eine Fehlermeldung ausgegeben. Eingefügt werden können alle DIM-Knoten/Objekte (mit Ausnahme des Projekt-Knotens).

3.7 Undo/Redo-Funktionalität

Undo (englisch für zurücknehmen, rückgängig machen) ist die Bezeichnung für die Funktion des Datenbasis-Editors, eine oder mehrere Benutzereingaben zurückzunehmen. Die Undo-Funktion ist verknüpft mit dem "Redo", d.h. der Möglichkeit, nach einem Rückgängigmachen dieselben Schritte ohne Neueingabe nochmals ausführen zu lassen. Dem Undo/Redo-Mechanismus unterliegen das Hinzufügen, das Ändern und das Löschen von DIM-Objekten.

Menüpunkt	Funktion
Undo	Macht eine Änderung rückgängig (Ctrl-Z)
Redo	Nimmt das Rückgängigmachen einer Änderung zurück (Ctrl-Y)

3.7.1 Undo-Menüpunkt

Das Ausführen des Undo-Menüpunktes bewirkt, dass die zuletzt ausgeführten Änderungen am markierten DIM-Knoten rückgängig gemacht werden. Rückgängig gemacht werden können das Löschen, das Einfügen und das Bearbeiten eines gewählten Datensatzes.

3.7.2 Redo-Menüpunkt

Das Ausführen des Redo-Menüpunktes bewirkt, dass die Änderung, die zuletzt rückgängig gemacht wurde erneut ausgeführt wird. Erneut ausgeführt werden können das Löschen, das Einfügen und das Bearbeiten eines gewählten Datensatzes.

3.8 Drag and Drop-Funktionalität

Drag and Drop (englisch drag and drop = Ziehen und Fallenlassen) ist eine Methode zur Bedienung des Datenbasis-Editors durch das Bewegen von DIM-Objekten mit der Maus. Ein DIM-Knoten kann damit gezogen und über einem möglichen Ziel-Knoten losgelassen werden. Im Allgemeinen kann Drag and Drop genutzt werden, um Kopier-Aktionen zwischen DIM-Objekten auszuführen.

3.8.1 Markieren des Elementes

Da Drag als Ziehen mit gedrückter Maustaste definiert ist, wird die Operation durch das Drücken der linken Taste eingeleitet, wobei sich der Mauscursor über einem DIM-Knoten befindet. Die Auswahl des Elements markiert den Beginn des Drag-and-Drop-Vorgangs.

3.8.2 Visualisieren des Ziehens

Der Datenbasis-Editor gibt eine visuelle Rückmeldung sobald ein DIM-Element mit der Maus bei gedrückt gehaltener linker Maustaste verschoben wird. Ein halbtransparentes Abbild des ausgewählten Elementes wird mit dem Mauscursor mitbewegt und so die Drag-and-Drop-Möglichkeit angezeigt.

3.8.3 Anzeige einer Drop-Möglichkeit

Wird ein DIM-Element über ein mögliches Drop-Ziel gezogen, so ist dies an dem gezogenen Element erkennbar. Bewegt sich das Objekt in einem Bereich, der nicht für Drop geeignet ist, so wird dies durch einen Hinweis in Form eines Verbotsschilds dargestellt.

3.8.4 Drop (Loslassen)

Nach dem Drop (engl. loslassen oder fallen lassen) eines DIM-Elementes erfolgt eine Rückmeldung über den Erfolg der Aktion. Das neue DIM-Objekt wird innerhalb der Baum-Hierarchie eingefügt.

3.9 Quick Start - Definition einer Beispieldatenbasis

Wir wollen nun mit Hilfe des Datenbasis-Editors eine Daten-Interpreter-Datenbasis erstellen. Mit dieser soll das Protokoll eines Gerätes beschrieben werden, welches die folgenden Eigenschaften haben soll:

- Das Gerät verfügt über einen Schalter mit drei Stellungen, einen Drucksensor und einen Temperatursensor
- Nach dem Einschalten überträgt das Gerät zunächst die Werte seiner Prozessvariablen (Schalterstellung, Druck und Temperatur) zyklisch über den CAN-Bus. Der verwendete CAN Identifier sei 5
- Die Schalterstellung wird im ersten Datenbyte als Unsigned Integer übertragen und kann die Werte 0,1 und 2 (für Aus, Mittelstellung und Vollast) annehmen

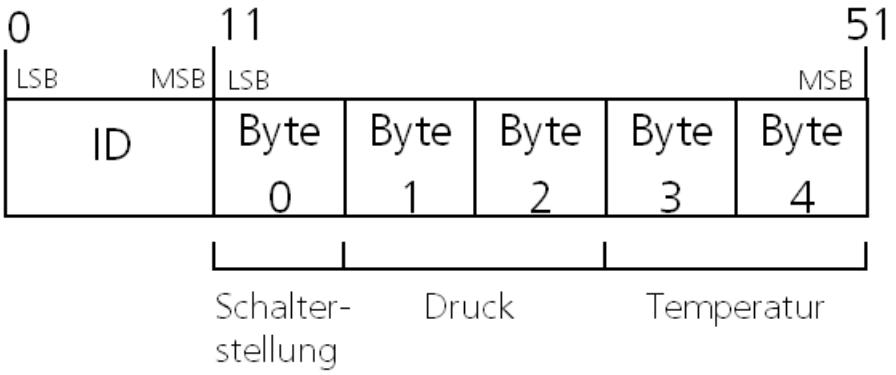


Abbildung 3.11: Beispieltelegramm

- Die nächsten beiden Bytes enthalten den Druck in hPa im INTEL-Format als DWORD ohne Vorzeichen
- In Byte 3 und 4 wird die Temperatur in °C im INTEL-Format als vorzeichenbehaftetes DWORD übertragen

Die Einordnung (Mapping) der zu übertragenden Prozessdaten in das CAN-Telegramm zeigt Abb. 3.11.
Mit Hilfe des Datenbasis-Editors soll nun eine Datenbasis erstellt werden, mit dem dieses CAN-Telegramm interpretiert werden kann.

3.9.1 Anlegen eines neuen Projekts

Nach dem Start des Datenbasis-Editors wird automatisch ein leeres Projekt angelegt. Es kann auch jederzeit ein neues Projekt mit Hilfe des Befehls **File|New** angelegt werden.
In einem leeren Projekt ist nur der Projekt-Knoten enthalten. Auf der rechten Seite werden die Projektdaten angezeigt und können geändert werden (Abb. 3.12).

3.9.2 Anlegen eines Nachrichtenobjekts für den CAN-Identifier

Durch die erstellte Daten-Interpreter-Datenbasis soll nur das Telegramm mit dem ID 5 interpretiert werden. Dazu wird ein Nachrichtenobjekt erstellt.
Klicken Sie mit der rechten Maustaste auf den Projektknoten. Das Popup-Menu aus Abb. 3.13 erscheint.
Fügen Sie nun eine Nachricht ein, indem Sie den Menüpunkt **Add Message** wählen.
Nach dem Anwählen des neu erstellten Message-Knotens erhalten Sie eine leere Eingabemaske.
Definieren Sie nun den Identifier für die auszuwertende Nachricht mit "5". Außerdem können Sie der Nachricht einen Namen zuweisen (Abb. 3.14).

3.9.3 Definition von Signalen

Nachdem Sie nun ein Nachrichtenobjekt für den CAN-Identifier definiert haben, müssen Sie die Signale festlegen, die im CAN-Telegramm mit ID=5 enthalten sind.

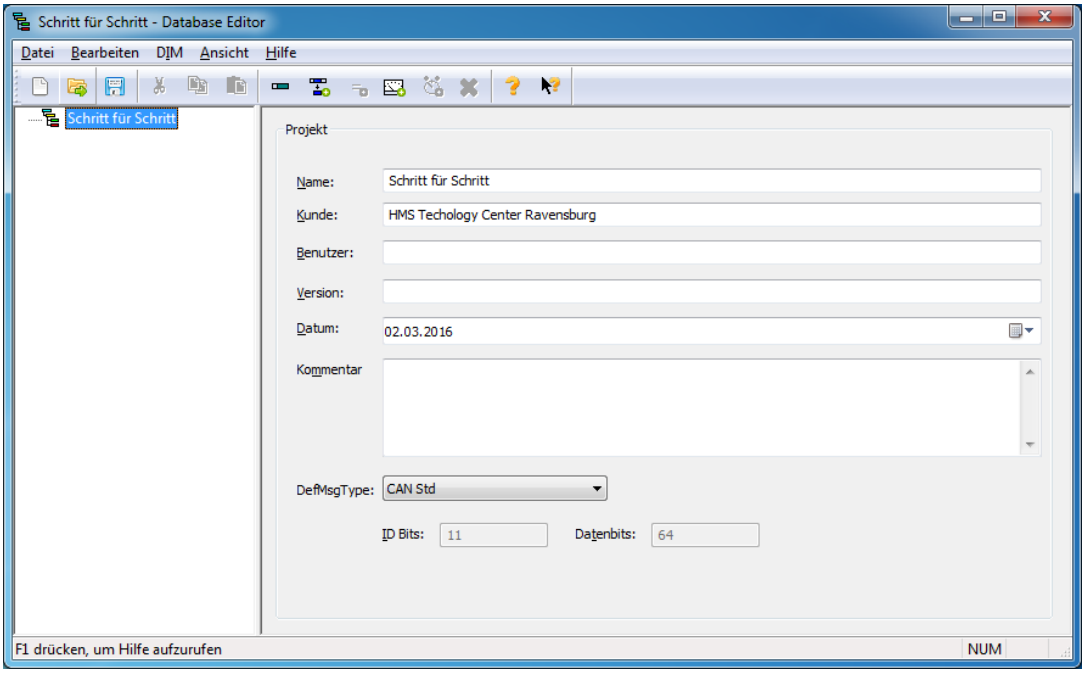


Abbildung 3.12: Eingabe der Projektdaten

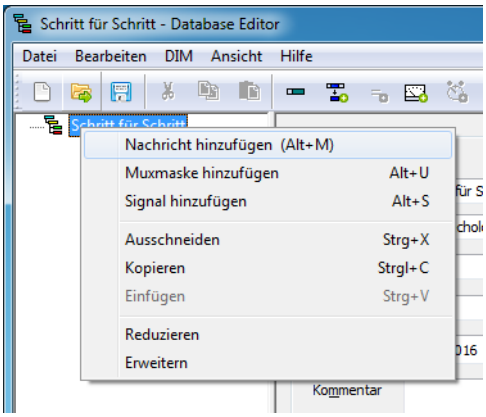


Abbildung 3.13: Einfügen eines Nachrichtenobjekts

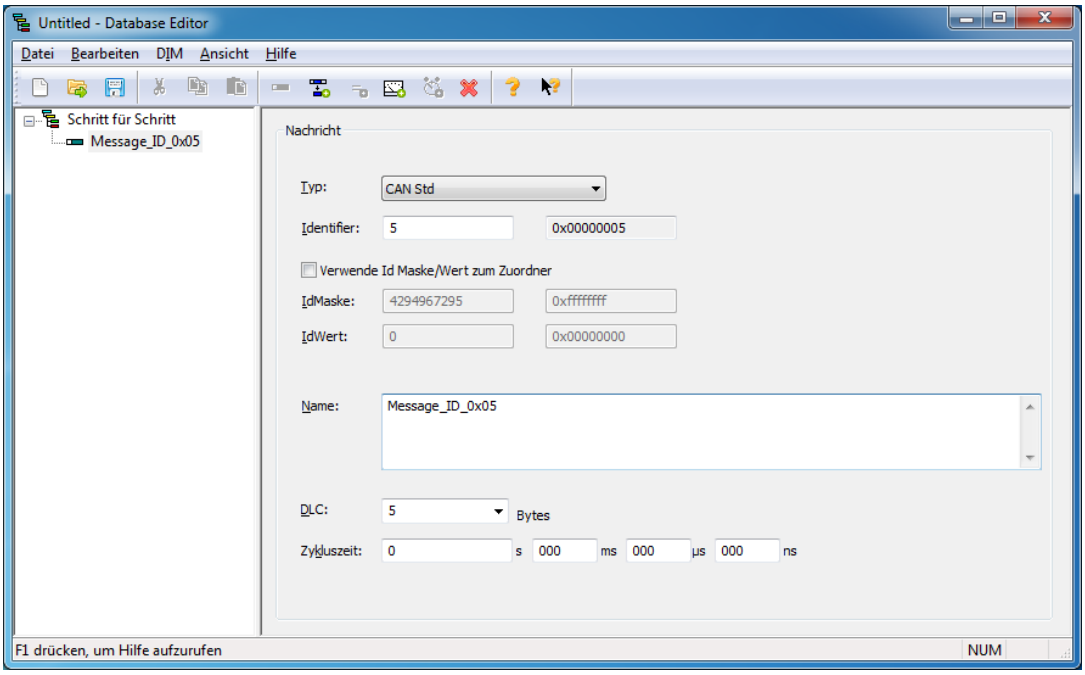


Abbildung 3.14: Einfügen eines Nachrichtenobjekts

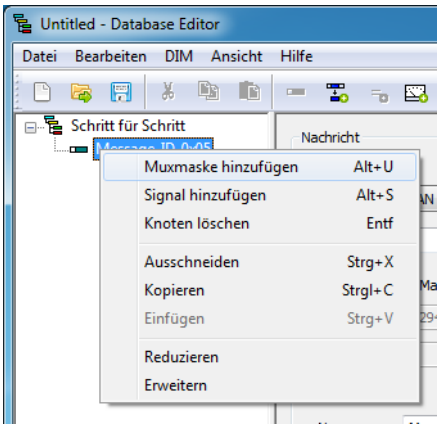


Abbildung 3.15: Hinzufügen eines Signaldefinition

Fügen Sie dazu unterhalb des Nachrichtenobjekts eine erste Signaldefinitionen ein. Dazu klicken Sie mit der rechten Maustaste auf das Nachrichtenobjekt und erhalten das Kontextmenü aus Abb. 3.15.
Fügen Sie nun eine Signaldefinition ein indem Sie den Menüpunkt **Add Signal** wählen.
Zunächst einmal legen Sie die Attribute für das Signal "Druck" fest. Entsprechend unserer Annahme soll der Druckwert in Byte 2 und Byte 3 des Datenfelds übertragen werden. Die Einheit ist hPa, der Maximaldruck wurde auf 10000 hPa festgelegt. Der Druckwert wird eins zu eins übertragen, deshalb wird Scale auf den Wert 1 und Offset auf den Wert 0 gesetzt (Abb. 3.16).
Den Quelldatentyp haben wir auf ein DWORD im INTEL-Format ohne Vorzeichen festgelegt. Machen Sie die entsprechenden Einstellungen in den Feldern "Datatype" und "Encoding".
Im Attribut "Display" können Sie die Anzeige des Signalwerts durch das Signal-Modul parametrieren.
"Scale" und "Offset" sind zwei Parameter einer Geradengleichung anhand derer die Umrechnung der Rohdaten auf den physikalischen Wert des Signals erfolgt.
Über "MinValue", "MaxValue" und "DefaultValue" kann eine Überprüfung bzw. Vorbelegung der Signalwerte erfolgen.

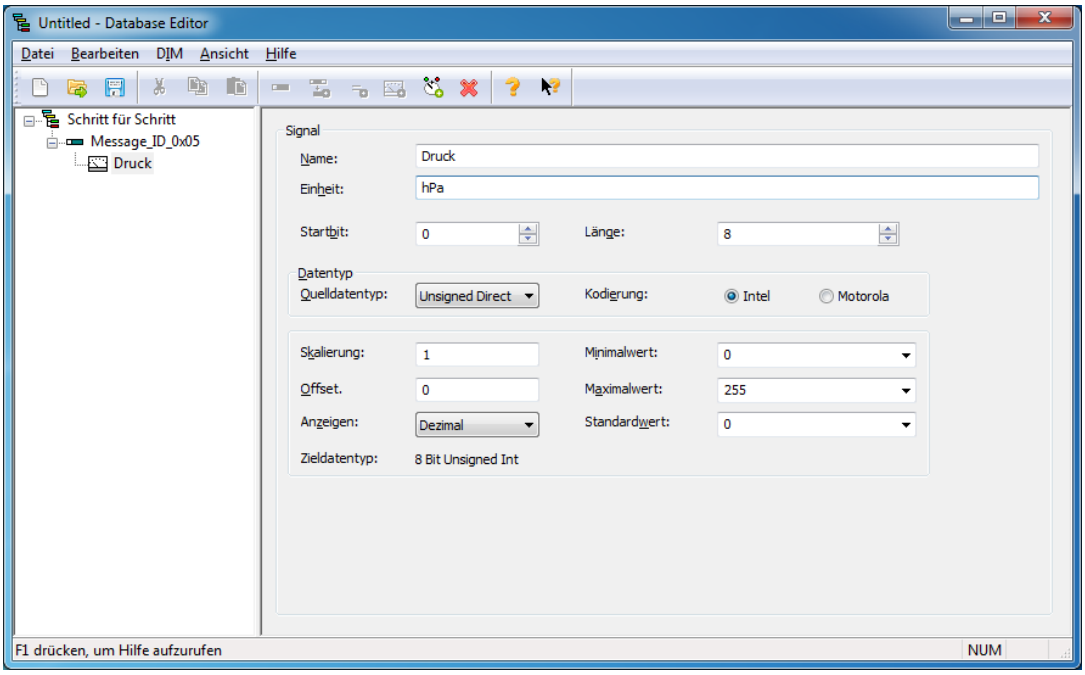


Abbildung 3.16: Definition des Signals "Druck"

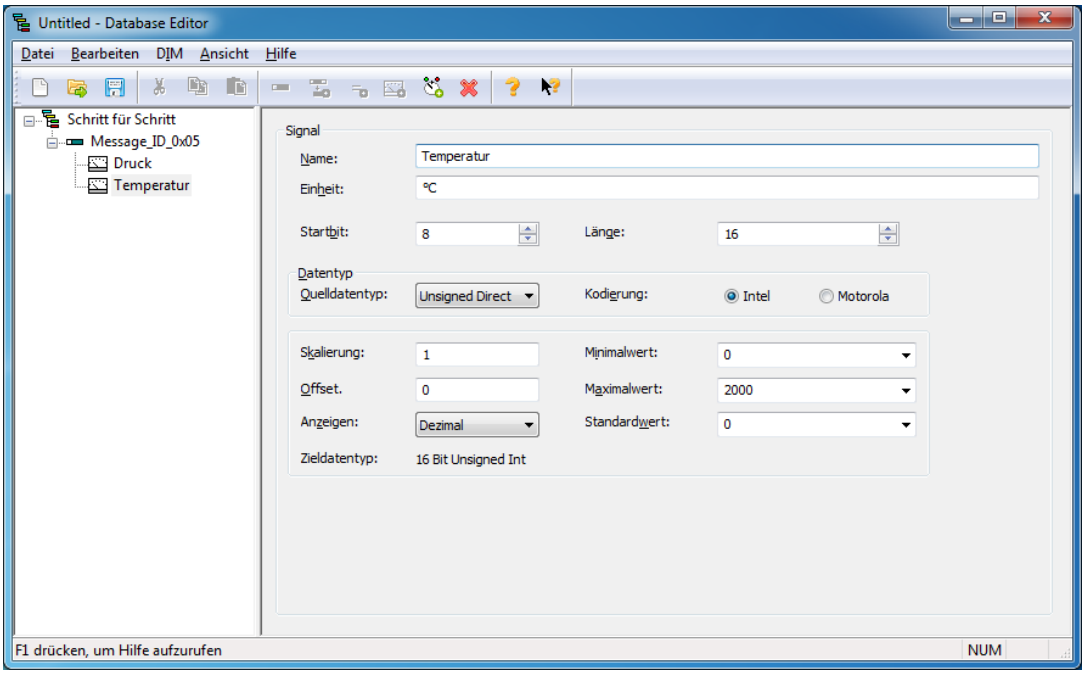


Abbildung 3.17: Definition des Signals "Temperatur"

Anschließend legen Sie das Signal "Temperatur" an. Sie wird in Byte 3 und 4 des Datenfelds abgelegt und besitzt einen Maximalwert von 2000 °C. Die Temperatur wird eins zu eins übertragen, deshalb wird Scale auf den Wert 1 und Offset auf den Wert 0 gesetzt (Abb. 3.17).
Der Quelldatentyp ist in diesem Fall ein vorzeichenbehaftetes DWORD im INTEL-Format.
Zum Schluss legen Sie noch das digitale Signal "Schalter" an. Der Zustand des Schalters befindet sich in Byte 0 des Datenfelds und wird als vorzeichenloser ganzzahliger Wert übertragen (Abb. 3.18).
Bei dem Signal "Schalter" handelt es sich um ein digitales Signal, welches Zustände beschreibt. Der Schalter verfügt über die drei Stellungen bzw. Zustände (in Klammern sind die zugeordneten

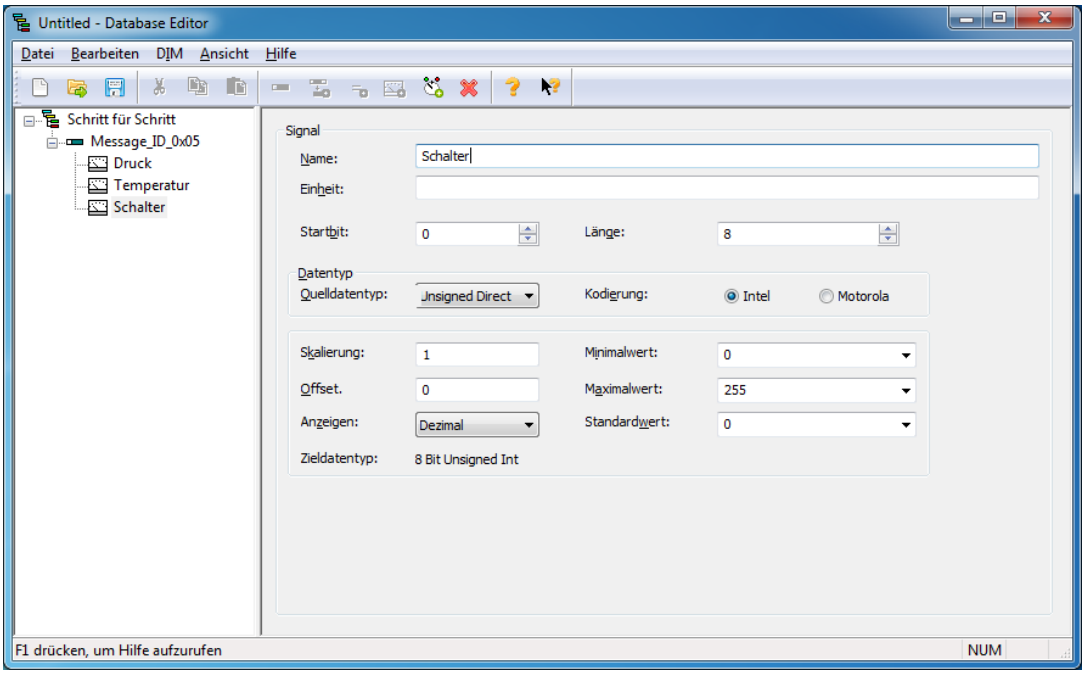


Abbildung 3.18: Definition des Signals "Schalter"

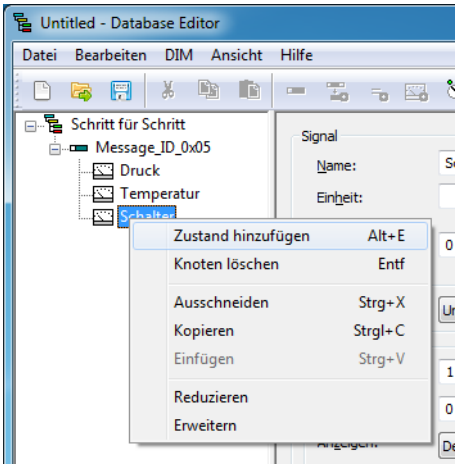


Abbildung 3.19: Hinzufügen eines Zustands

numerischen Werte angegeben):

- Aus (0)
- Mittelstellung (1)
- Vollast (2)

3.9.4 Definition von Zuständen

Fügen Sie nun die Definition für die drei Zustände zur Datenbasis hinzu.
Dazu klicken Sie mit der rechten Maustaste auf den Signal-Knoten "Schalter" und Sie erhalten das Kontextmenu aus Abb. 3.19.
Fügen Sie nun einen State-Datensatz ein indem Sie den Menüpunkt **Add State** wählen.
Nach Auswahl des neu angelegten State-Datensatzes können Sie den Namen und den numerischen Wert des Zustands eingeben (Abb. 3.20).

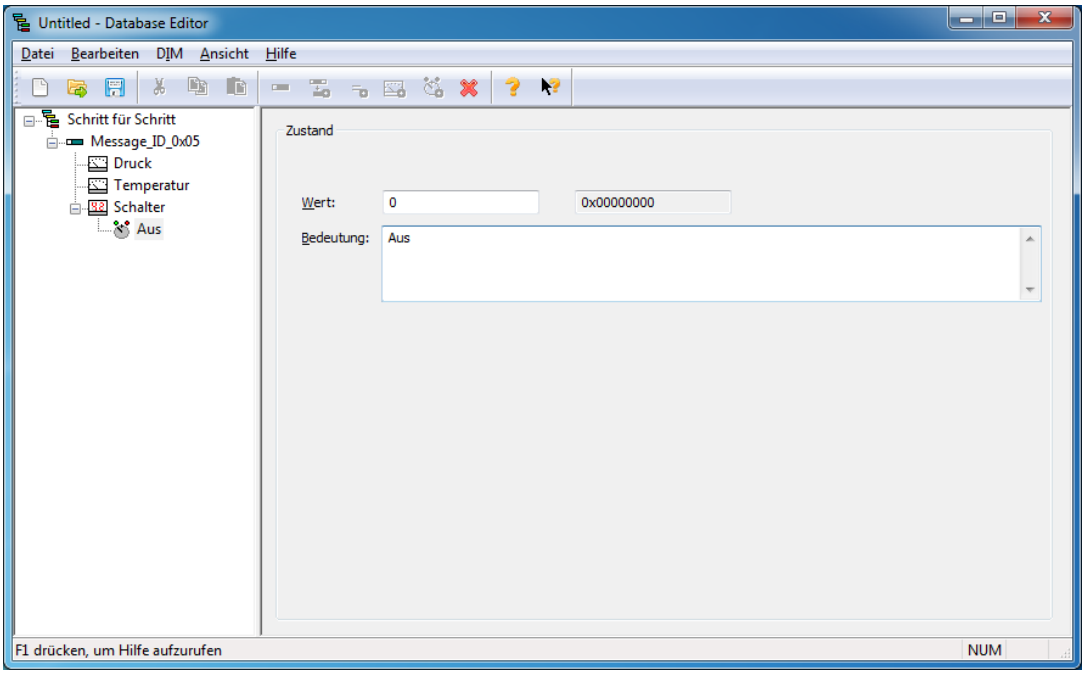


Abbildung 3.20: Definition des Zustands "Aus"

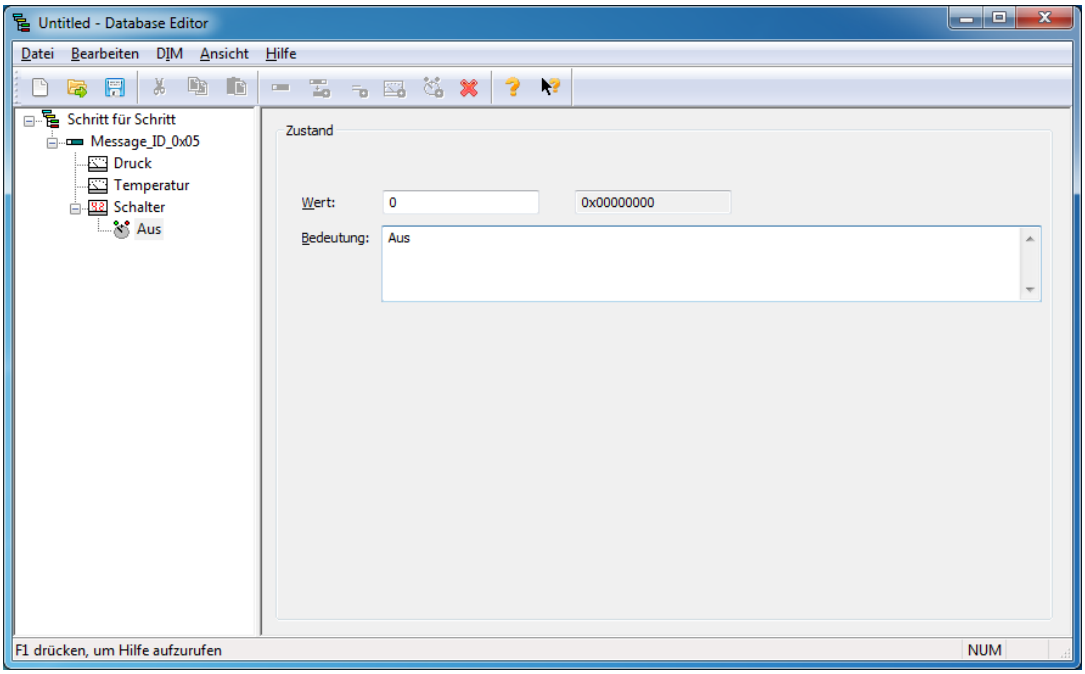


Abbildung 3.21: Der vollständige Projektbaum

Für die verbleibenden zwei Zustände "Mittelstellung" und "Vollast" gehen Sie analog vor, so dass Sie am Ende den in Abb. 3.21 dargestellten Projektbaum erhalten.
Damit ist die Definition der für unser Beispiel erforderlichen Datenbasis abgeschlossen.

Kapitel 4

Verfügbare Importfilter

Derzeit sind die folgenden Importfilter verfügbar:

- Eigenes XML-Format (Import und Export)
- DCF-Import (gemappte PDOs)
- CANDB-Import (Fa. Vector)

4.1 DCF-Import (*.dcf)

Der DCF-Import wird benutzt, um aus einer CANopen DCF-Datei gemappte PDOs zu extrahieren. Dies wird vor allem dazu benutzt, um den OPC-Server von IXXAT mit Hilfe des CANopen Configuration Studios konfigurieren zu können.

4.2 CANDB-Import (*.dbc)

Eine weitere Importmöglichkeit besteht für CAN-DB Dateien. Dabei werden nur die für die Interpretation relevanten Daten aus bestehenden DBC-Dateien entnommen, und damit eine Daten-Interpreter-Datenbasis generiert.
Weitere in CANDB-Dateien enthaltene Informationen wie Environmentvariablen und benutzerdefinierte Attribute werden vom CANDB-Importfilter ignoriert.

Kapitel 5

Einschränkungen

- Im Daten-Interpreter-Modul wird die Position der Daten im Bitstrom durch die Angabe von Anfangsbit und Bitlänge beschrieben. Das Daten-Interpreter-Modul kann dadurch keine Informationen verarbeiten, die auf mehrere unzusammenhängende Bereiche in einem Bitstrom verteilt sind.
- Zustände können allen Signalen zugeordnet werden, jedoch ist die Zuordnung von Zuständen nur für Signale des Datentyps "vorzeichenlose Binärzahl" sinnvoll.
- Remote Request Nachrichten werden durch das Daten-Interpreter-Modul nicht ausgewertet. Dies ist jedoch keine wesentliche Einschränkung, da RTR-Frames keine Daten enthalten.

Kapitel 6

Dateiformat

6.1 Einleitung

Die Projektdaten des Datenbasis-Editors werden in einer Projektdatei gespeichert, die dem XML-Standard entspricht. Dieser Anhang beschreibt die zur Strukturierung der Datei verwendeten Elemente (Tags) und das Format der abgelegten Daten.

6.2 Struktur

6.2.1 Header

Jede XML-Datei, also auch die Daten-Interpreter-Projektdatei, beginnt mit der folgenden Zeichenfolge:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DIM-PROJECT SYSTEM 'dimprj.dtd'>
```

Hier wird die XML-Version angegeben, zu der die Datei konform ist, sowie die Kodierung der enthaltenen Zeichen. Das !DOCTYPE-Tag gibt den Namen des Starttags (DIM-PROJECT) sowie eine evtl. vorhandene DTD-Datei (Document-Type-Definition) an.

6.2.2 Start-Tag

Das Starttag für eine Projektdatei ist <DIM-PROJECT>. Dieses Tag beinhaltet alle für ein Daten-Interpreter-Projekt relevanten Daten.

```
<DIM-PROJECT>

...weitere Daten des DIM-Projekts

</DIM-PROJECT>
```

6.2.3 Projektdaten

Innerhalb des <DIM-PROJECT>-Tags werden die Projektdaten mit Hilfe des Tags <PRJ-DATA> zusammengefasst. Es darf nur einmal innerhalb des Projekts vorhanden sein.

```
<DIM-PROJECT>
  <PRJ-DATA>
```

```
    <PRJ-NAME>Demo-Projekt</PRJ-NAME>
    <PRJ-CUSTOMER>IXXAT Automation GmbH</PRJ-CUSTOMER>
    <PRJ-USER>U.R.</PRJ-USER>
    <PRJ-VERSION>1</PRJ-VERSION>
    <PRJ-COMMENT>Comment</PRJ-COMMENT>
    <PRJ-DATE>27/11/2003</PRJ-DATE>
    <PRJ-FRAMEFORMAT>0xb</PRJ-FRAMEFORMAT>
    <PRJ-NUMIDBITS>0xb</PRJ-NUMIDBITS>
    <PRJ-NUMDATABITS>0x40</PRJ-NUMDATABITS>
  </PRJ-DATA>

  ...keine, eine oder mehrere Nachrichten

</DIM-PROJECT>
```

Die einzelnen Tags trennen die Projektdaten und haben folgende Bedeutung:

Tag	Bedeutung
PRJ-NAME	Projektname
PRJ-CUSTOMER	Firma
PRJ-USER	Benutzer
PRJ-VERSION	Version
PRJ-COMMENT	Kommentar (max 1024 Zeichen)
PRJ-DATE	Datum
PRJ-FRAMEFORMAT	Standard frame format beim Anlegen neuer MESSAGE Beschreibungen
PRJ-NUMIDBITS	Anzahl Identifierbits
PRJ-NUMDATABITS>	Anzahl Datenbits

Für das Attribut PRJ-FRAMEFORMAT sind folgende Konstanten zulässig:

constant	Meaning
0x02	DIM_FF_CAN11BIT
0x03	DIM_FF_CAN29BIT
0x05	DIM_FF_FLEXRAY
0x06	DIM_FF_CANFD_STD
0x07	DIM_FF_CANFD_EXT

6.2.4 Nachrichtendaten

Eine Nachricht beschreibt die zu interpretierende Nachricht durch den CAN-Identifier. Eine Nachrichten-Definition erfolgt durch das Tag <Message>, die Daten sind durch das Tag <MSG-DATA> gekennzeichnet.

```
<MESSAGE>
  <MSG-DATA>
    <MSG-NAME>MyMessage</MSG-NAME>
    <MSG-LENGTH>0x8</MSG-LENGTH>
    <MSG-ID>0x1a0</MSG-ID>
    <MSG-IDMASK>0xfffffff0</MSG-IDMASK>
    <MSG-IDVALUE>0x200</MSG-IDVALUE>
```

```
      <MSG-FRAMEFORMAT>0x2</MSG-FRAMEFORMAT>
      <MSG-CYCLETIME>0</MSG-CYCLETIME>
      <MSG-STATUS>0x7</MSG-STATUS>
    </MSG-DATA>

    ...höchstens eine MuxMaske
    ...eine oder mehrere multiplexerunabhängige Signale

  </Message>
```

Die einzelnen Tags trennen die Projektdaten und haben folgende Bedeutung:

Tag	Bedeutung
MSG-NAME	Nachrichtenname
MSG-LENGTH	Nachrichtenlänge
MSG-ID	Nachrichtenidentifier
MSG-FRAMEFORMAT	Nachrichtenformat. Dieselben Konstanten wie bei PRJ-FRAMEFORMAT sind zulässig.
MSG-CYCLETIME	Zykluszeit
MSG-STATUS	Status

Versionen ab 02/2016 fügten folgende Tags hinzu, um Mask/Wert-Paare beim zuordnen des Nacrichtenidentifiers zu unterstützen:

Tag	Bedeutung
MSG-IDMASK	ID-Maske
MSG-IDVALUE	ID-Vergleichswert

Das Zuordnen einer Nachrichtenbeschreibung wird nach folgendem Algorithmus durchgeführt:

```
if (0xFFFFFFFF == MSG-IDMASK) then
  ismatch = (msg.id == MSG-ID)
else
  ismatch = ((msg.id & MSG-IDMASK) == MSG-IDVALUE)
end
```

6.2.5 MuxMasken

Eine MuxMasken-Definition wird durch das Tag <MUXMASK> eingeschlossen. Die Daten der MuxMaske sind durch das Tag <MM-DATA> gekennzeichnet.

```
<MUXMASK>
  <MM-DATA>
    <MM-NAME>Standard CAN-Identifier<MM-NAME>
    <MM-STARTBIT>0x15</MM-STARTBIT>
    <MM-BITLENGTH>0xb</MM-BITLENGTH>
  </MM-DATA>

  ...kein, ein  oder mehrere MuxValues
  ...höchstens ein ElseValue

</MUXMASK>
```

Die einzelnen Tags trennen die Daten der MuxMaske:

Tag	Bedeutung
MM-NAME	Name der Muxmaske
MM-STARTBIT	Anfangsbit
MM-BITLENGTH	Anzahl Bits

6.2.6 MuxValues

Innerhalb einer Muxmasken-Definition können sich keine, eine oder mehrere MuxValue-Definitionen befinden.

```
<MUXMASK>
  <MM-DATA>
    [Daten]
  </MM-DATA>
  <MUXVALUE>
    <VAL-DATA>
      <VAL-NAME>ID_100</VAL-NAME>
      <VAL-VALUE>100</VAL-VALUE>
      <VAL-STATUS>0x7</VAL-STATUS>
    </VAL-DATA>

    ...keine, eine oder mehrere Signale

  </MUXVALUE>

...weitere MuxValues
...höchstens ein ElseValue

</MUXMASK>
```

Die einzelnen Tags trennen die Daten eines MuxValue:

Tag	Bedeutung
VAL-NAME	Name
VAL-VALUE	Wert
VAL-STATUS	Status

6.2.7 ElseValues

Ein ElseValue stellt einen Alternativzweig dar und besitzt deshalb nur die Attribute Name und Status.

```
...
<ELSEVALUE>
  <VAL-DATA>
    <VAL-NAME>else</VAL-NAME>
    <VAL-STATUS>0x7</VAL-STATUS>
  </VAL-DATA>
```

```
    ...keine, eine oder mehrere Signale

</ELSEVALUE>
...
```

Die einzelnen Tags haben die gleiche Bedeutung wie bei normalen MuxValues. Das Tag <VALVALUE> wird innerhalb eines ElseValues ignoriert.

6.2.8 Signale

Signal sind die zentrale Struktur für das Daten-Interpreter-Modul. Sie werden in das Tag <VARIABLE> eingeschlossen:

```
...
<VARIABLE>
  <VAR-DATA>
    <VAR-NAME>Var_1</VAR-NAME>
    <VAR-STARTBIT>0x20</VAR-STARTBIT>
    <VAR-BITLENGTH>0x8</VAR-BITLENGTH>
    <VAR-UNIT></VAR-UNIT>
    <VAR-DEFVALUE>0</VAR-DEFVALUE>
    <VAR-SCALE>1</VAR-SCALE>
    <VAR-OFFSET>0</VAR-OFFSET>
    <VAR-MINVAL>0</VAR-MINVAL>
    <VAR-MAXVAL>255</VAR-MAXVAL>
    <VAR-STATUS>0x7</VAR-STATUS>
    <VAR-TYPE>0x48010</VAR-TYPE>
    <VAR-DISPCOLOR>0</VAR-DISPCOLOR>
    <VAR-FORMATSTR></VAR-FORMATSTR>
  </VAR-DATA>

  ...kein, ein oder mehrere States

</VARIABLE>
...
```

Folgende Tags trennen Daten einer Signaldefinition:

Tag	Bedeutung
VAR-NAME	Name
VAR-STARTBIT	Startbit
VAR-BITLENGTH	Bitlänge
VAR-UNIT	Einheit
VAR-DEFVALUE	Default-Wert
VAR-SCALE	Skalierung
VAR-OFFSET	Offset
VAR-MINVAL	Minimum-Wert
VAR-MAXVAL	Maximum-Wert
VAR-STATUS	Status
VAR-TYPE	Quelltyp
VAR-DISPCOLOR	Farbe
VAR-FORMATSTR	Formatstring

VAR-TYPE ist ein 32 Bit Wert, der den Quelltyp der Signale kodiert. Er hat folgenden Aufbau:

Bit	Bedeutung
0 - 7	Datatype:0x10 = Direkt; 0x11 = 1er Kompl.; 0x12 = 2er Kompl.; 0x13 = BCD; 0x20 = IEEE; 0x30 = String
8 - 14	reserviert
15	Encoding:0x0 = Intel; 0x1 = Motorola
16 - 23	Display Format:0x00 = FormatString; 0x01 = Ascii; 0x02 = Binär; 0x03 = Octal; 0x04 = Dezimal; 0x05 = Hexadezimal
24 - 30	reserviert
31	Vorzeichen:0x0 = unsigned; 0x1 = signed

6.2.9 States

States (Zustände) sind unterhalb von Variablendefinitionen angeordnet.

```
[...]
<VARIABLE>
  <VAR-DATA>
    [...]
  </VAR-DATA>
  <STATE>
    <STATE-DATA>
      <STATE-MEANING>Aus</STATE-MEANING>
      <STATE-VALUE>0</STATE-VALUE>
    </STATE-DATA>
  </STATE>
  [weitere States]
</VARIABLE>
[...]
```

Folgende Tags gehören zu einem State-Datensatz:

Tag	Bedeutung
STATE-MEANING	Bedeutung des Zustands
STATE-VALUE	Wert